NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

FIXED POINT SMOOTHING ALGORITHM TO THE TORPEDO TRACKING PROBLEM

by

Sadi Karaman

JUN 1986

Thesis Advisor: H. A. TITUS

Approved for public release; distribution unlimited.

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY 2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (If applicable) 62	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000
8a NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO | PROJECT NO | TASK NO | WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
FIXED POINT SMOOTHING ALGORITHM TO THE TORPEDO TRACKING PROBLEM

12 PERSONAL AUTHOR(S) Sadi Karaman

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM ______ TO ______	1986, June	

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Fixed Point Smoothing, Torpedo Tracking, Sequential Extended Kalman Filter

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

A sequential extended Kalman filter and optimal smoothing algorithm was developed to provide real time estimates of torpedo position and depth on the three dimensional underwater tracking range at the Naval Torpedo Station Keyport, Washington. The measurements consisted of acoustic pulse transit times from the torpedo to the receiving array, which are nonlinear functions of the positions and the depth of the torpedo, were linearized and filter gains and filtered estimates of states calculated. By running the smoothing subroutine, all past filtered estimates of states and error covariance were smoothed. The program was tested, using simulated torpedo trajectories that traversed both single and multiple arrays, on an IBM-PC. The results showed that filter performance was dependent on system noise and the distance to the hydrophone array from the torpedo and the smoothed estimates of states and error covariances were better than or equal to the filtered estimates.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Prof. Harold A. Titus	22b TELEPHONE (Include Area Code) (408) 646-2560 | 22c OFFICE SYMBOL [illegible]

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

Approved for public release; distribution is unlimited.

Fixed Point Smoothing Algorithm to the Torpedo Tracking Problem.

by

Sadi Karaman
LTJG., Turkish Navy
B.S., Turkish Naval Academy, 1979

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1986

Author: ______________________
Sadi Karaman

Approved by: ______________________
H.A. Titus, Thesis Advisor

____
Alex Gerba, Second Reader

____
H.B. Rigas, Chairman, Department of
Electrical and Computer Engineering

____
J.N. Dyer, Dean of Science
and Engineering

ABSTRACT

A sequential extended Kalman filter and optimal smoothing algorithm was developed to provide real time estimates of torpedo position and depth on the three dimensional underwater tracking range at the Naval Torpedo Station, Keyport, Washington. The measurements consisted of acoustic pulse transit times from the torpedo to receiving array, which are nonlinear functions of the positions and the depth of the torpedo, were linearized and filter gains and filtered estimates of states calculated. By running the smoothing subroutine, all past filtered estimates of states and error covariance were smoothed. The program was tested, using simulated torpedo trajectories that traversed both single and multiple arrays, on an IBM-PC. The results showed that filter performance was dependent on system noise and the distance to the hydrophone array from the torpedo and the smoothed estimates of states and error covariances were better than or equal to the filtered estimates.

A1

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENT

I would like to express my gratitude to Professor Hal Titus for his professional guidance, assistance and encouragement during the course of this research. I would also like to thank to Professor Alex Gerba for his help and suggestions.

Finally, I want to thank my wife, Maria, for her patience and support, and my father Sait and my mother Zeynep from whom I inherited the desire for education.

I. INTRODUCTION

The Naval Torpedo Station at Keyport, Washington currently operates two three-dimensional underwater tracking ranges utilizing a sonar transmitter installed in the torpedo to be tracked. The transmitter is synchronized with a master clock. Timed acoustic pulses are received by hydrophone arrays and then relayed via cable to a computer at the observation site. The computer calculates the positional coordinates of the torpedo and plots its trajectory through the water.

The measured data, which consist of the elapsed time from transmission of a pulse until its receipt at the hydrophone array, is corrupted with noise due to combined effects of environmental factors and measurement instruments.

The intention is to implement and test a sequential extended Kalman filter and smoothing routine which processes the transit times of the acoustic pulses and generates the filtered and smoothed estimates of the positions of tracked torpedo at a particular time. The design takes into account the elimination of the storage problem.

II. DESCRIPTION OF RANGE TRACKING GEOMETRY

The hydrophone array, consisting of four independent elements, defines an orthogonal coordinate system in which transit time measurements are made. As shown in Figure 2.1, four hydrophones X, Y, Z, and C are on four adjacent vertices separated by a distance d, along the edge of the cube. The origin of the array coordinates is at the center of the cube with the orthogonal coordinates parallel to its edge. Positional information is computed from the transit times of a periodic synchronous acoustic signal traveling from the torpedo to the four hydrophones on the array. The torpedoes are equipped with sonar transmitters which are transmitting an acoustic signal in every 1.31 seconds, within a range accuracy 3 to 30 ft. When tracking by multiple arrays, the signal from the closest hydrophone array is defined as the basis for the time measurements and for the range calculations. A more detailed description of the range tracking capability is described in [Ref. 1, 2].

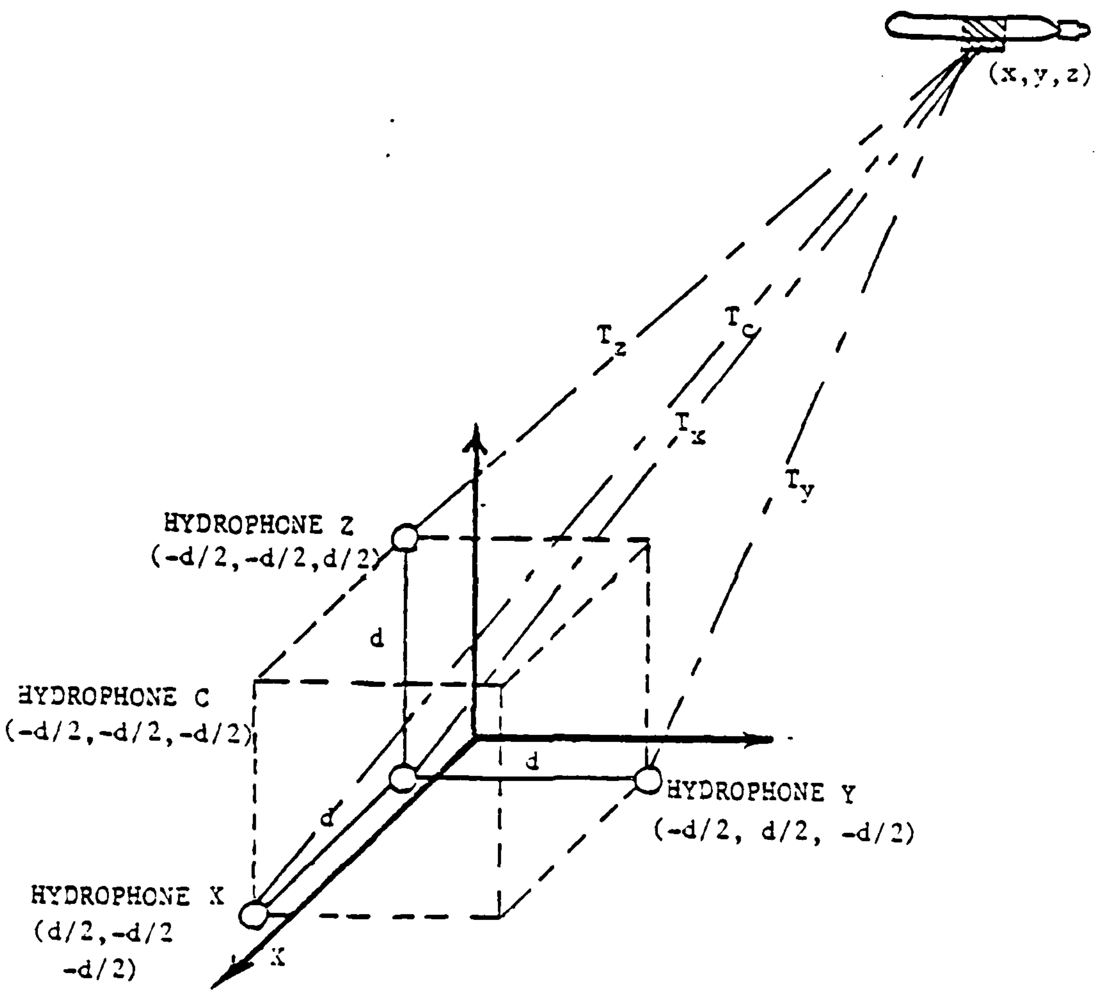


Figure 2.1 Geometry of a Tracking Array

III. THEORY

A. EXTENDED KALMAN FILTER

The basic idea of the extended Kalman filter is to relinearize about each estimate $\hat{\underline{X}}(k/k)$, once it has been computed. As soon as a new state estimate is made, a new and better reference state trajectory is incorporated into the estimation process. [Ref. 3, 4, 5]

For the three-dimensional location problem three position states (X, Y, Z) and two velocity states (V_x , V_y) specify target motion. The discrete linear and nonlinear observation equations are given by

$$\underline{X}(k+1) = \Phi \cdot \underline{X}(k) + \Gamma \cdot \underline{W}(k) \qquad (3.1)$$

and

$$\underline{Z}(k) = \underline{h}(\underline{X}(k), k) + \underline{V}(k) \qquad (3.2)$$

where: Φ and Γ are constant matrices;

$\underline{h}$ is a nonlinear function of the state $\underline{X}$

$\underline{W}(k)$ is the plant excitation noise;

$\underline{V}(k)$ is the measurement noise.

In these equations the plant noise and measurement noise are assumed uncorrelated (white) with zero mean. That is,

$$E[\underline{W}(k) . \underline{W}^T(j)] = Q'(k) \ \delta_{kj}$$

and

$$E[\underline{V}(k).\underline{V}^T(j)] = R(k)\ \delta_{kj}$$

where: $\delta = 1, \quad k = j;$

$= 0, \quad k \neq j.$

In order to apply the linear filter, Equation 3.2 is expanded in a Taylor series about the best estimate of the state at that time and only the first order terms are kept. Equation 3.2 gives

$$\underline{Z}(k) = H(k)\ .\ \underline{X}(k) + \underline{V}(k) \tag{3.3}$$

where

$$H(k) = \left.\frac{\partial \underline{h}}{\partial \underline{X}}\right|_{\underline{X}(k) = \hat{\underline{X}}(k/k-1)} \tag{3.4}$$

$\hat{\underline{X}}(k/k - 1)$ is a predicted value of the state at time k, given the measurements until time k-1.

A state error vector is defined by

$$\tilde{\underline{X}}(k/k) = \hat{\underline{X}}(k/k) - \underline{X}(k),$$

and a predicted state error vector is defined by

$$\tilde{\underline{X}}(k/k - 1) = \hat{\underline{X}}(k/k - 1) - \underline{X}(k).$$

The covariance of state error matrix is defined by

$$P(k/k) = E[\tilde{\underline{X}}(k/k).\tilde{\underline{X}}^T(k/k)],$$

the predicted covariance of state error matrix is given by

$$P(k/k - 1) = E[\tilde{\underline{X}}(k/k - 1).\tilde{\underline{X}}^T(k/k - 1)].$$

The state excitation matrix is given by

$$Q(k) = \Gamma . E[\underline{W}(k) . \underline{W}^T(k)] . \Gamma^T,$$

and the measurement noise covariance matrix is

$$R(k) = E[\underline{V}(k) . \underline{V}^T(k)].$$

The Kalman filter equations are given by [Ref. 3, 4, 5]:

$$P(k+1/k) = \Phi \; P(k/k) \; \Phi^T + Q(k) \tag{3.5}$$

$$G(k) = P(k/k-1) H^T(k) [H(k) P(k/k-1) H^T(k) + R(k)]^{-1} \tag{3.6}$$

$$P(k/k) = [I - G(k) \; H(k)] \; P(k/k-1) \tag{3.7}$$

$$\hat{\underline{X}}(k+1/k) = \Phi \; \hat{\underline{X}}(k/k) \tag{3.8}$$

$$\hat{\underline{Z}}(k/k-1) = \underline{h}(\hat{\underline{X}}(k/k-1), \; k) \tag{3.9}$$

$$\hat{\underline{X}}(k/k) = \hat{\underline{X}}((k/k-1) + G(k) \; [\underline{Z}(k) - \hat{\underline{Z}}(k/k-1)] \tag{3.10}$$

The Q matrix serves not only to allow for maneuvering but also to account for any model inaccuracies, that is, any discrepancies between the true action of the torpedo and its characterization by Equation 3.1. The Q matrix also serves to prevent the gain matrix G(k) from approaching zero by always insuring uncertanity in the predicted covariance of error matrix P(k+1/k) [Ref. 1, 3, 4, 5].

B. OPTIMAL SMOOTHING

Smoothing is a non-real time data processing scheme that uses all measurements between 0 and N to estimate the state of a system at certain time k, where $0 \leq k \leq N$. The smoothed estimate of $\underline{X}(k)$ based on all measurements between 0 and N is denoted by $\hat{\underline{X}}(k/N)$. The smoothed error covariance is denoted by $P(k/N)$ and $P(k/N) \leq P(k/k)$ means that the smoothed estimate of $\underline{X}(k)$ is at least as good as the filtered estimate or equal to its filtered estimate for all the time, except the terminal time. This is shown graphically in Figure 3.1. As portrayed in Figure 3.2, there are three classes of particular interest because of their applicability to realistic problems [Ref. 3, 4, 5]. One is the Rauch-Tung-Striebel form, which was chosen in our particular problem [Ref. 6, 7].

The smoothed state estimate and the smoothed error covariance matrix are given by

$$\hat{\underline{X}}(k/N) = \hat{\underline{X}}(k/k) + A(k)[\hat{\underline{X}}(k+1/N) - \hat{\underline{X}}(k+1/k)] \qquad (3.11)$$

$$\hat{\underline{X}}(k+1/k) = \underline{\Phi}\,\hat{\underline{X}}(k/k) \qquad (3.12)$$

$$P(k/N)=P(k/k)+A(k)[P(k+1/N)-P(k+1/k)]A(k)^T \qquad (3.13)$$

where

$$A(k) = P(k/k)\,\underline{\Phi}^T\,P^{-1}(k+1/k) \qquad \text{for } k \leq N.$$

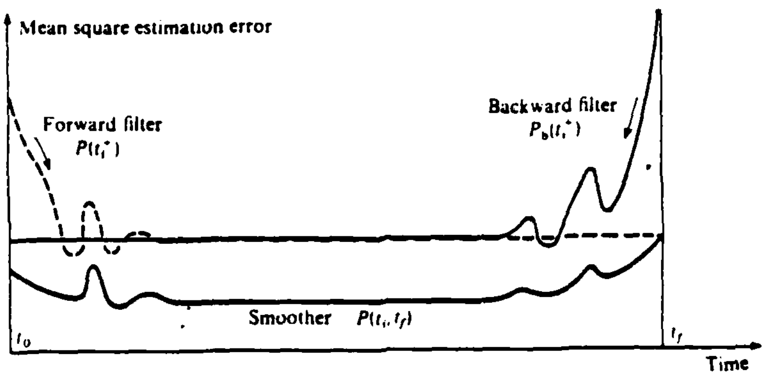


Figure 3.1 Advantage of Performing Optimal Smoothing

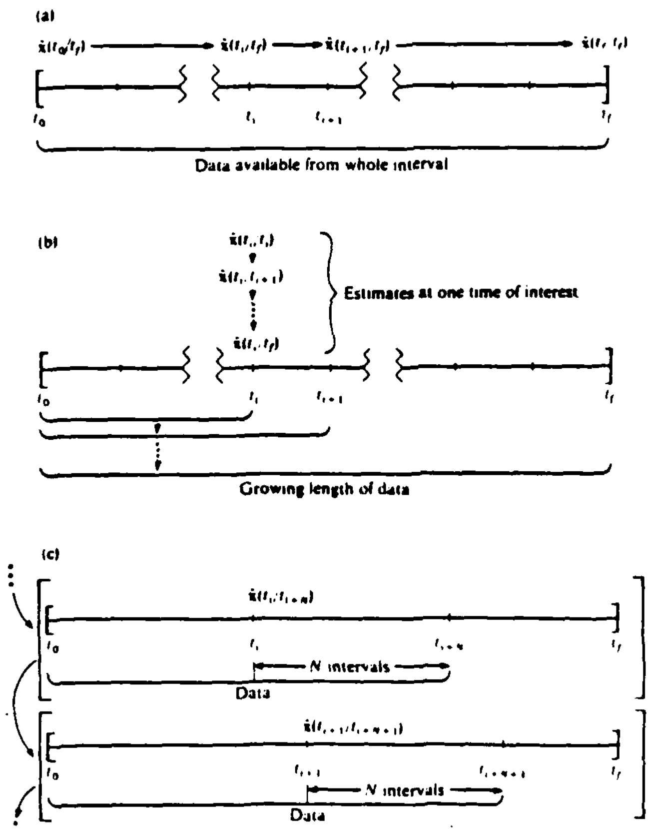


Figure 3.2 Three types of smoothing: (a) fixed-interval, (b) fixed-point, (c) fixed-lag smoothing.

IV. PROBLEM DEFINITION

A. OBSERVATION AND PLANT STATE EQUATIONS

In the torpedo tracking problem, the non-linear observation equations are the four independent transit times from the tracked torpedo to the hydrophones, T_c, T_x, T_y and T_z. Thus the non-linear measurement matrix is defined by

$$\underline{Z}(k) = [T_c(k) \quad T_x(k) \quad T_y(k) \quad T_z(k)]^T + \underline{V}(k) \qquad (4.1)$$

where

$$T_c(k) = \frac{1}{vel}[(X(k)+d/2)^2 + (Y(k)+d/2)^2 + (Z(k)+d/2)^2 \quad]^{1/2}$$

$$T_x(k) = \frac{1}{vel}[(X(k)-d/2)^2 + (Y(k)+d/2)^2 + (Z(k)+d/2)^2 \quad]^{1/2}$$

$$T_y(k) = \frac{1}{vel}[(X(k)+d/2)^2 + (Y(k)-d/2)^2 + (Z(k)+d/2)^2 \quad]^{1/2}$$

$$T_z(k) = \frac{1}{vel}[(X(k)+d/2)^2 + (Y(k)+d/2)^2 + (Z(k)-d/2)^2 \quad]^{1/2}$$

Since the transit times are readily available and non-linear functions of position, these equations can be linearized and Kalman filter theory applied using the extended Kalman filter. This procedure produces a real time

filtering on the transit times T_c, T_x, T_y and T_z, without the necessity of converting these times to positions.

Equation 3.4 can be used to give the linearized observation matrix. When the derivatives are taken and evaluated at the predicted state values $\hat{\underline{X}}(k/k-1) = \underline{X}^*(k)$ the result is

$$H(k) = \frac{1}{vel} \begin{vmatrix} \frac{X^*(k) + d/2}{den1} & 0 & \frac{Y^*(k) + d/2}{den1} & 0 & \frac{Z^*(k) + d/2}{den1} \\ \frac{X^*(k) - d/2}{den2} & 0 & \frac{Y^*(k) + d/2}{den2} & 0 & \frac{Z^*(k) + d/2}{den2} \\ \frac{X^*(k) + d/2}{den3} & 0 & \frac{Y^*(k) - d/2}{den3} & 0 & \frac{Z^*(k) + d/2}{den3} \\ \frac{X^*(k) + d/2}{den4} & 0 & \frac{Y^*(k) + d/2}{den4} & 0 & \frac{Z^*(k) - d/2}{den4} \end{vmatrix}$$

where:

$$den1 = [(X^*(k)+d/2)^2 + (Y^*(k)+d/2)^2 + (Z^*(k)+d/2)^2]^{1/2}$$

$$den2 = [(X^*(k)-d/2)^2 + (Y^*(k)+d/2)^2 + (Z^*(k)+d/2)^2]^{1/2}$$

$$den3 = [(X^*(k)+d/2)^2 + (Y^*(k)-d/2)^2 + (Z^*(k)+d/2)^2]^{1/2}$$

$$den4 = [(X^*(k)+d/2)^2 + (Y^*(k)+d/2)^2 + (Z^*(k)-d/2)^2]^{1/2}$$

The measurement noises, $V(k)$'s, are assumed to be zero-mean and independent with a covariance matrix

$$R(k) = \begin{vmatrix} \sigma^2_{T_c} & 0 & 0 & 0 \\ 0 & \sigma^2_{T_x} & 0 & 0 \\ 0 & 0 & \sigma^2_{T_y} & 0 \\ 0 & 0 & 0 & \sigma^2_{T_z} \end{vmatrix}$$

The plant state equations are

$$\begin{vmatrix} X(k+1) \\ V_x(k+1) \\ Y(k+1) \\ V_y(k+1) \\ Z(k+1) \end{vmatrix} = \begin{vmatrix} X(k) + T\,V_x(k) + g_1 \\ V_x(k) + g_2 \\ Y(k) + T\,V_y(k) + g_3 \\ V_y(k) + g_4 \\ Z(k) + g_5 \end{vmatrix} \qquad (4.2)$$

where $X(k)$, $Y(k)$ and $Z(k)$ are the position coordinates of the torpedo at time $t(k)$, $V_x(k)$ and $V_y(k)$ are the X and Y components of the velocity.

The excitation terms g_1 through g_5 are included to take into account the random changes in speed (γ_{V_t}), heading (γ_{θ_t}), and depth (γ_Z), which are assumed to be independent, zero mean, rates of changes. Typical maneuvering parameters for the torpedo are given in [Ref. 8].

$$\sigma_{\theta_t} = 22\ ^{o}/\text{sec}; \qquad \sigma^2_{\theta_t} = E[\gamma^2_{\theta_t}]$$

$$\sigma_{V_t} = 36\ \text{ft/sec}^2; \qquad \sigma^2_{V_t} = E[\gamma^2_{V_t}]$$

$$\sigma_Z = 1\ \text{ft / sec}; \qquad \sigma^2_Z = E[\gamma^2_Z]$$

The effect of this excitation is to increase the predicted covariance of the state error matrix.

The excitation covariance matrix is given by

$$Q = \Gamma . E[\underline{W}(k)\ \underline{W}^T(k)] . \Gamma^T \tag{4.3}$$

and

$$\sigma^2_{\dot{X}} = \left(\frac{V_x}{V_t}\right)^2 \sigma^2_{V_t} + V_y^2\ \sigma^2_{\theta_t}$$

$$\sigma^2_{\dot{Y}} = \left(\frac{V_Y}{V_t} \right)^2 \sigma^2_{\dot{V}_t} + V_x^2 \, \sigma^2_{\dot{\theta}_t}$$

$$\sigma_{\dot{X}\cdot\dot{Y}} = V_x \, V_Y \left[\frac{\sigma^2_{\dot{V}_t}}{V_t^2} - \sigma^2_{\dot{\theta}_t} \right]$$

where the states are evaluated at the current state estimates $\hat{\underline{X}}(k/k)$. Substituting these expressions in the Q matrix results in

$$Q = \begin{vmatrix} \left(\frac{T^2}{2}\right)^2 \sigma^2_{\dot{X}} & \frac{T^3}{2}\sigma^2_{\dot{X}} & \left(\frac{T^2}{2}\right)^2 \sigma_{\dot{X}\cdot\dot{Y}} & \frac{T^3}{2}\sigma_{\dot{X}}\cdot\sigma_{\dot{Y}} & 0 \\ & T^2 \sigma^2_{\dot{X}} & \frac{T^3}{2}\sigma_{\dot{X}\cdot\dot{Y}} & T^2 \sigma_{\dot{X}\cdot\dot{Y}} & 0 \\ & & \left(\frac{T^2}{2}\right)^2 \sigma^2_{\dot{Y}} & \frac{T^3}{2}\sigma^2_{\dot{Y}} & 0 \\ & & & T^2 \sigma^2_{\dot{Y}} & 0 \\ \text{symmetric} & & & & T^2 \sigma^2_{Z} \end{vmatrix}$$

A more detailed derivation of the excitation covariance matrix is given in [Ref. 8].

The excitation matrix serves not only to take into account the possibility of maneuvering, but of model inaccuracies as well. Q also used to prevent the gains of

the filter from approaching zero as more and more data is processed, by insuring some uncertainty in the predicted state values [Ref. 3, 4, 5].

In the state form, the plant state equation is

$$\underline{X}(k+1) = \Phi\, \underline{X}(k) + \Gamma\, \underline{W}(k) \qquad (4.4)$$

where:

$$\Phi = \begin{vmatrix} 1 & T & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & T & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{vmatrix} \quad \text{and} \quad \Gamma = \begin{vmatrix} T^2/2 & 0 & 0 \\ T & 0 & 0 \\ 0 & T^2/2 & 0 \\ 0 & T & 0 \\ 0 & 0 & 0 \end{vmatrix}$$

B. DEFINITION OF MULTIPLE ARRAY TRACKING

The coordinate system is defined as shown in Figure 4.1. These 72 positions, an XYZ position for each of 4 hydrophones in 6 arrays, are placed into a 6 x 12 matrix HYDRO and referenced throughout the program. The torpedo position is referenced to a central level rectangular coordinate system. The non-linear observation equations become

$$\underline{Z}(k) = [T_C(k) \quad T_x(k) \quad T_y(k) \quad T_z(k)]^T + \underline{V}(k) \qquad (4.5)$$

where

$$T_C(k) = \frac{1}{vel}[(X(k)-X_{iC})^2 + (Y(k)-Y_{iC})^2 + (Z(k)-Z_{iC})^2\ ]^{1/2}$$

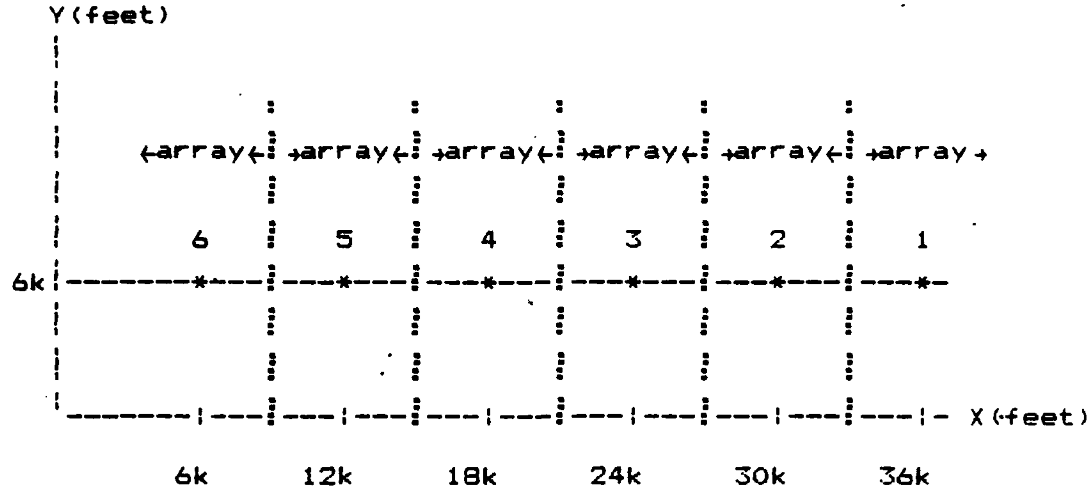
$$T_x(k)=\frac{1}{vel}[(X(k)-X_{iX})^2+(Y(k)-Y_{iX})^2+(Z(k)-Z_{iX})^2]^{1/2}$$

$$T_y(k)=\frac{1}{vel}[(X(k)-X_{iY})^2+(Y(k)-Y_{iY})^2+(Z(k)-Z_{iY})^2]^{1/2}$$

$$T_z(k)=\frac{1}{vel}[(X(k)-X_{iZ})^2+(Y(k)-Y_{iZ})^2+(Z(k)-Z_{iZ})^2]^{1/2}$$

and the subscripted variables X, Y, and Z are the coordinates of a particular array being used.

The decision parameter used to determine the switching from array to array is a straight handoff. If the predicted x position, $\hat{X}_{k+1/k}$, is greater than 3,000 feet from the array in use, then index is incremented and the next row of HYDRO is implememted. This placed into the program the X, Y, and Z positions of the hydrophones in the next array. The handoff can easily be utilized in real range operations, as the transit times from adjacent arrays are present at the computer for a particular time slot.



a) Coordinate System for Multiple Array Tracking

C HYDRO x	C HYDRO y	C HYDRO z	X HYDRO x	X HYDRO y	X HYDRO z	Y HYDRO x	Y HYDRO y	Y HYDRO z	Z HYDRO x	Z HYDRO y	Z HYDRO z
36000	6000	0	36030	6000	0	36000	6030	0	36000	6000	30
30000	6000	0	30030	6000	0	30000	6030	0	30000	6000	30
24000	6000	0	24030	6000	0	24000	6030	0	24000	6000	30
18000	6000	0	18030	6000	0	18000	6030	0	18000	6000	30
12000	6000	0	12030	6000	0	12000	6030	0	12000	6000	30
6000	6000	0	6030	6000	0	6000	6030	0	6000	6000	30

b) Hydrophone Array Location Matrix

Figure 4.1 Geometry of Multiple Array Tracking

C. SEQUENTIAL EXTENDED KALMAN FILTER

In the sequential approach, after modifying the basic Kalman filter equations, calculations are performed on each of the four independent transit times in the following order: T_c, T_x, T_y and T_z for each 1.31 second time slot. Since the four transit times are independent and processed sequentially, the covariance of error matrix and the state vector are updated four times during each time slot. Thus more accurate estimates of the filter states are achived. Modification of the filter equations for the sequential approach circumvented the matrix inversion in the gain equation. An invalid transit time measurement will result in the filter ignoring the update information for that particular measurement only.

The estimate of the states $\underline{\hat{X}}(k/k)$, based on one transit time measurement are used as the prediction $\underline{\hat{X}}(k/k-1)$ for the calculationson the next measurements. Thus for the first time measurement T_c only the first row of the linearized H matrix is calculated and then the first gain column corresponding to the first time measurement T_c is calculated by

$$G_{icol} = \frac{P(k/k-1) \quad H^T_{irow}}{H_{irow} \quad P(k/k-1) \quad H^T_{irow} + R_{ii}} \tag{4.6}$$

where $i = 1$ to 4 corresponding to the four measured transit times.

An estimate of the particular observation time is calculated by using Equation 3.9 evaluated at the predicted state $\hat{\underline{X}}(k/k-1)$. The difference between observed transit times and the estimated transit times forms the residual which is used in the estimate equation

$$\hat{\underline{X}}_i = \hat{\underline{X}}(k/k-1) + G_{icol} \text{ [Residual]} \tag{4.7}$$

This equation gives an estimate of the states based on one of the four time measurements.

The covariance of error is calculated based on one measurement by

$$P_i = [I - G_{icol} \ H_{irow}] \ P_{i-1} \tag{4.8}$$

where: I is identity matrix;

P_{i-1} is the covariance matrix calculated from the previous transit time measurement or if $i = 1$, the predicted error covariance $P(k/k-1)$.

Editing erroneous time measurement is achieved by implementing a three sigma gate using the covariance of the measurent noise (R) and the covariance of the estimation

error $P(k/k)$.The gate then is written for each time measurement $i = 1$ to 4:

$$\text{gate} = 3* \{[(P_{jj}\text{maximum})/(4860.)^2] + R_{ii}\}^{1/2} \quad (4.9)$$

where $j = 1, 3, 5$. The gate expands or decreases depending on the confidence level of the position estimate and the transit time. If the difference between the actual transit time received and predicted transit time to a particular hydrophone exceeds the gate, the measurement is considered unacceptable and the filter gain is set to zero causing the filter to ignore the data and take the prediction of the states as the estimate $\hat{\underline{X}}(k/k) = \hat{\underline{X}}(k/k-1)$.

Bounding the residual bias error is achieved by making comparison between the average of the absolute value of the time residuals and the preset threshold. If the average of the time residuals exceeds the preset threshold, Q is calculated and added to the last updated covariance of error matrix P. Then filter reiterates the gain, covariance, and state estimate equations for the same time slot. This procedure continues until the average of the time residuals falls below the preset threshold at which time an acceptable state vector estimate has been obtained for the time slot.

D. OPTIMAL SMOOTHING ALGORITHM

The smoothing solution starts with the filtered estimate at the last point and calculates backward point by point determining the smoothed estimate as a linear combination of the filtered estimate at that point and the smoothed estimate at the previous point [Ref. 6].

It can be seen from the error covariances that the filter has reached a steady-state condition by the end of the forward sweep. As an example, let us enter the backward sweep at the end point where k = 20. Here we have $\hat{\underline{X}}(20/20)$ and P(20/20). Since the filter solution at this point is conditioned on all the measurement data, it is also the smoothed estimate at k = N = 20. We are now ready to compute the smoothed estimate one step back at k = 19. From Equations 3.11, 3.12 and 3.13 we have

$$\hat{\underline{X}}(19/20) = \underbrace{\hat{\underline{X}}(19/19)}_{\text{stored}} + A(19)[\underbrace{\hat{\underline{X}}(20/20)}_{\text{stored}} - \hat{\underline{X}}(20/19)]$$

$$\hat{\underline{X}}(20/19) = \Phi \underbrace{\hat{\underline{X}}(19/19)}_{\text{stored}}$$

$$A(19) = \underbrace{P(19/19)}_{\text{stored}} \Phi^T \underbrace{P^{-1}(20/19)}_{\text{stored}}$$

$$P(19/20) = \underset{\text{stored}}{P(19/19)} + A(19)[\underset{\text{stored}}{P(20/20)} - \underset{\text{stored}}{P(20/19)}]\, A^T(19)$$

and to compute the smoothed estimate two step back at k = 18

$$\hat{\underline{X}}(18/20) = \underset{\text{stored}}{\hat{\underline{X}}(18/18)} + A(18)[\hat{\underline{X}}(19/20) - \hat{\underline{X}}(19/18)]$$

$$\hat{\underline{X}}(19/18) = \Phi\, \underset{\text{stored}}{\hat{\underline{X}}(18/18)}$$

$$A(18) = \underset{\text{stored}}{P(18/18)}\, \Phi^T\, \underset{\text{stored}}{P^{-1}(19/18)}$$

$$P(18/20) = \underset{\text{stored}}{P(18/18)} + A(18)[P(19/20) - \underset{\text{stored}}{P(19/18)}]\, A^T(18)$$

This procedure continues until the time k reaches to 1.

V. SIMULATION RESULTS

A. MULTIPLE ARRAY ADAPTIVE MANEUVERING RUN

The true trajectory of the torpedo is a straight line with a 50 ft/sec velocity toward the origin of hydrophone array parallel to X-axis, drawing two tangent circles with 10 deg/sec turn rate, in the horizantal X-Y plane through a multiple array.

In the first part of this run, the initial position of the torpedo is 38000 ft in X, 7000 ft in Y, and 300 ft in Z. Figures 5.1 and 5.2 depict the filtered and smoothed estimate of the trajectory, with zero initial velocity errors and 25 ft initial position errors in X and Y. The errors in the filtered and the smoothed estimate of positions in X, Y and Z are drawn in Figures 5.3, 5.4, 5.5, 5.6, 5.7 and 5.8. For the Kalman filter, errors ranged between -1.2 and 2.6 ft in X, -5.9 and 1.9 ft in Y, 0.1 and 2.5 ft in Z. After smoothing, the errors occured in smaller range, which is, between -1.4 and 2.4 ft in X, -5.0 and 1.9 ft in Y, 0.1 and 0.7 ft in Z. The diagonal terms of the filtered and smoothed error covariance matrices are shown pictorially in Figures 5.9, 5.10, 5.11, 5.12, 5.13, 5.14, 5.15, 5.16, 5.17, 5.18.

In the second part of this run, the initial position of the torpedo is 35000 ft in X, 7000 ft in Y, and 300 ft in Z. The filtered and smoothed estimate of the trajectory are drawn in Figures 5.19 and 5.20. Taking this different initial geometry made the errors in the position of the torpedo to take place in bigger values during first time slot of the filtering and last time slot of the smoothing. As seen in Figures 5.21, 5.22, 5.23, 5.24, 5.25 and 5.26, errors ranged between -16.3 and 1.9 ft in X, -15.1 and 4.6 ft in Y, -5.0 and 1.0 ft in Z for the Kalman filter and for the smoothing this error range is between -12.6 and 1.3 ft in X, -12.3 and 4.8 ft in Y, -1.9 and 1.0 ft in Z. The diagonal terms of the filtered and smoothed error covariance matrices displayed slightly different magnitude, as seen in Figures 5.27, 5.28, 5.29, 5.30, 5.31, 5.32, 5.33, 5.34, 5.35 and 5.36.

B. MULTIPLE ARRAY ADAPTIVE STRAIGHT RUN

In this run, the true trajectory of the torpedo is a straight line with a 50 ft/sec velocity toward the origin of hydrophone array parallel to X-axis in the horizantal X-Y plane through a multiple array.

With the initial position of the torpedo is 38000 ft in X, 7000 ft in Y, and 300 ft in Z. The filtered and smoothed estimate of the trajectory, with zero initial velocity

errors and 25 ft initial position errors in X and Y, are depicted in Figures 5.37 and 5.38. Figures 5.39, 5.40, 5.41, 5.42, 5.43 and 5.44 give the errors in the filtered and the smoothed estimate of positions in X, Y and Z. For the Kalman filter, errors ranged between -1.6 and 2.6 ft in X, -5.9 and 4.7 ft in Y, -0.2 and 2.5 ft in Z. After smoothing, the errors occured in smaller range, which is, between -1.1 and 2.4 ft in X, -5.0 and 1.7 ft in Y, -0.2 and 0.6 ft in Z. The diagonal terms of the filtered and smoothed error covariance matrices are shown pictorially in Figures 5.45 through 5.54.

C. SINGLE ARRAY ADAPTIVE MANEUVERING RUN

The previous tests described the filter and smoothing performance for both straight and maneuvering runs through multiple array. Using the same basic torpedo trajectories as in multiple array, similar tests are performed for maneuvering run through single array. During the single array tracking, the initial position of the torpedo is 7500 ft in X, 1300 ft in Y and 0 ft in Z, which gives different initial geometry. The filtered and smoothed estimates of the trajectory and the corresponding position errors in X, Y and Z are pictorially given in Figures 5.55 through 5.62. For the Kalman filter, errors ranged between -1.6 and 3.3 ft in X, -19.1 and 8.9 ft in Y, -0.3 and 1.6 ft in Z. After smoothing, the errors occured in smaller range, which is,

between -1.1 and 3.1 ft in X, -17.9 and 5.0 ft in Y, -0.1 and 1.6 ft in Z. The diagonal terms of the filtered and smoothed error covariance matrices are shown pictorially in Figures 5.63 through 5.72.

D. SINGLE ARRAY STRAIGHT RUN

The purpose of this last series of tests is to functionally demonstrate the performance of the filter and smoothing during a straight run through single array using the same initial torpedo position as in single array adaptive maneuvering run. The filtered and smoothed estimates of the trajectory and the corresponding position errors in X, Y and Z are pictorially given in Figures 5.73 through 5.80. For the Kalman filter, errors ranged between -1.6 and 3.3 ft in X, -19.1 and 8.9 ft in Y, -0.3 and 0.8 ft in Z. After smoothing, the errors occured in smaller range, which is, between -0.6 and 3.1 ft in X, -17.9 and 3.5 ft in Y, -0.2 and 0.7 ft in Z. The diagonal terms of the filtered and smoothed error covariance matrices are shown pictorially in Figures 5.80 through 5.90.

VI. CONCLUSIONS

The sequential extended Kalman filter and smoothing routine sufficiently generated the filtered and smoothed estimates of the states, which specify the motion of the torpedo. Errors generated by running the routine on the IBM-PC are comparable to those given in the previous search, which was done on a large IBM computer [Ref. 1].

In the smoothing problem, computing the predicted estimates of the states, $\hat{\underline{X}}(k+1/k)$, from the estimates of the states, $\hat{\underline{X}}(k/k)$, eliminates the storage problem for $\hat{\underline{X}}(k+1/k)$. In future studies, an algorithm for computing $P(k/k)$ from $P(k+1/k+1)$ and hence eliminating the storage problem for $P(k/k)$, should be invesigated.

Examining the errors and their covariances, it is evident that the uncertainty in position exist only in the Y direction for the case where the torpedo is moving along the X axis. The results of the straight run analyses show that the propagation of the filtered error covariance is dependent on the path of the torpedo with respect to hydrophone array. Upon observing the error propagation it is apparent that the position errors exhibit approximately equal oscillations about zero indicating that the

measurement noise is the dominant error source driving the filter.

The smoothed estimates of the states are at least as good as or better than the filtered estimates. The filter performance was dependent on system noise and the distance from the torpedo to the hydrophone array. Errors get bigger as the torpedo approaches the tracking limit of the hydrophone array.

Additional work should be done using trajectories generated from actual torpedo runs on the Dabob test range. The rotation and reduction of the error ellipsoids should be also included in future studies.

The filter should be of use in range safety in warning for possible collisions. Also it may prove invaluable in torpedo recovery when there is a malfunction and the torpedo is sometimes buried in many feet of mude.

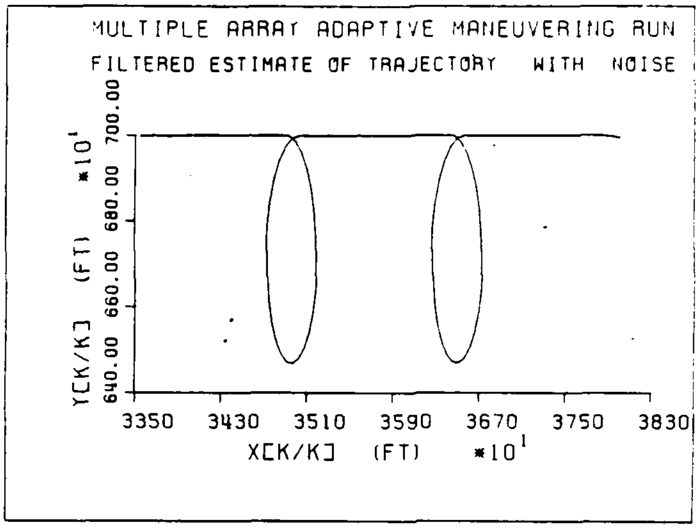


Figure 5.1 Filtered Estimate of Trajectory of the Torpedo During a Maneuvering Run through Multiple Array

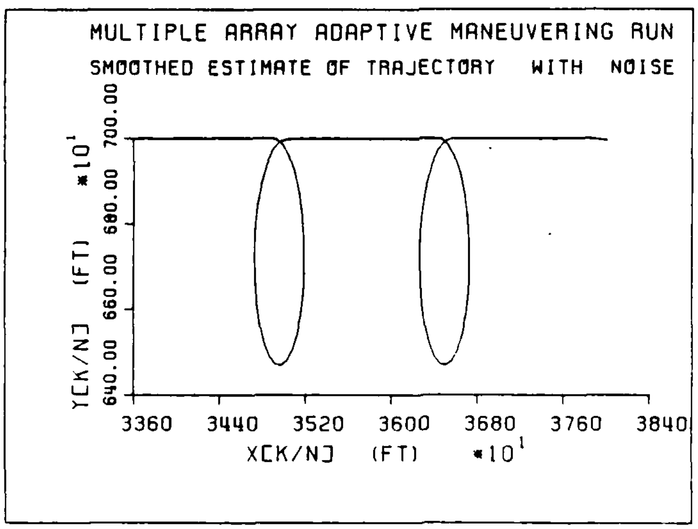


Figure 5.2 Smoothed Estimate of Trajectory of the Torpedo During a Maneuvering Run through Multiple Array

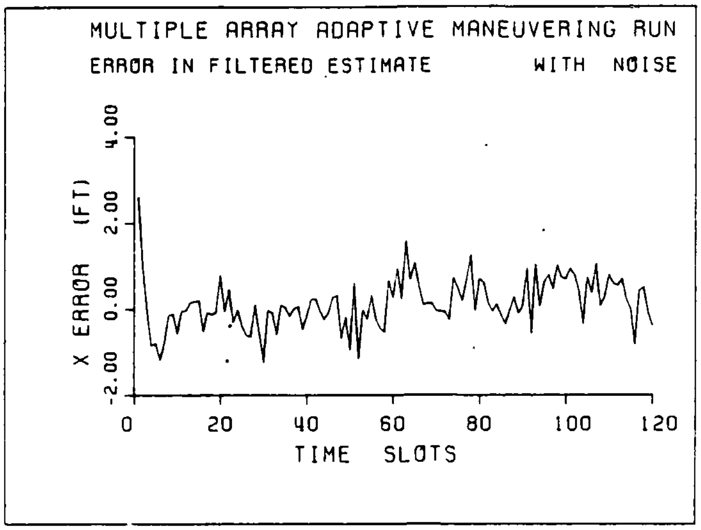


Figure 5.3 Error in Filtered Estimate of Position in X of the Torpedo During a Maneuvering Run through Multiple Array

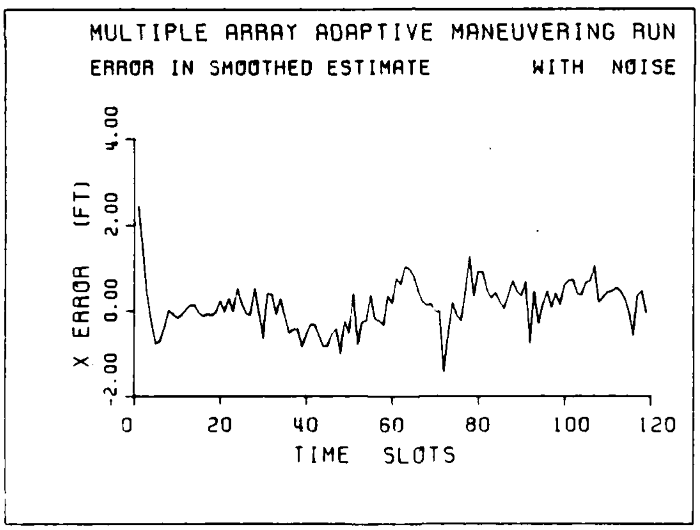


Figure 5.4 Error in Smoothed Estimate of Position in X of the Torpedo During a Maneuvering Run through Multiple Array

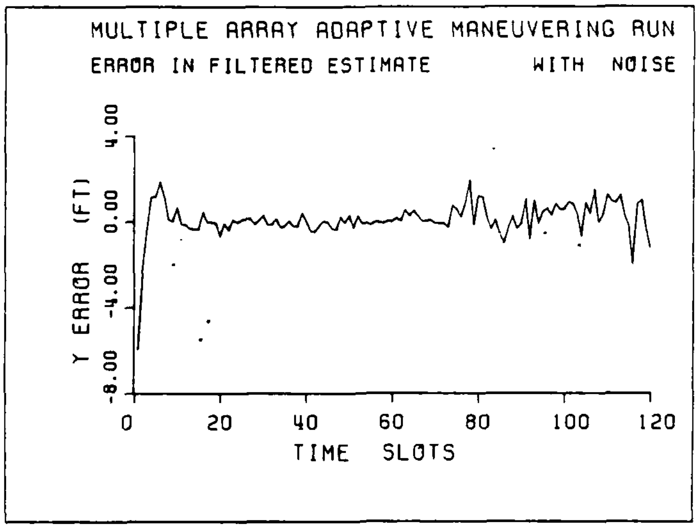


Figure 5.5 Error in Filtered Estimate of Position in Y of the Torpedo During a Maneuvering Run through Multiple Array

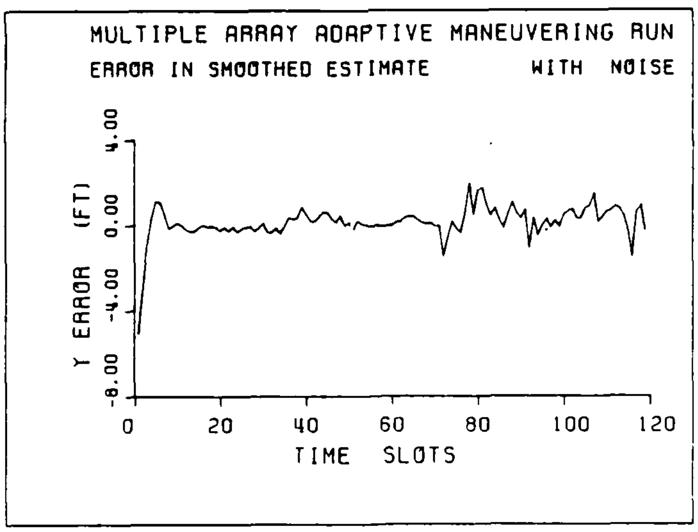


Figure 5.6 Error in Smoothed Estimate of Position in Y of the Torpedo During a Maneuvering Run through Multiple Array

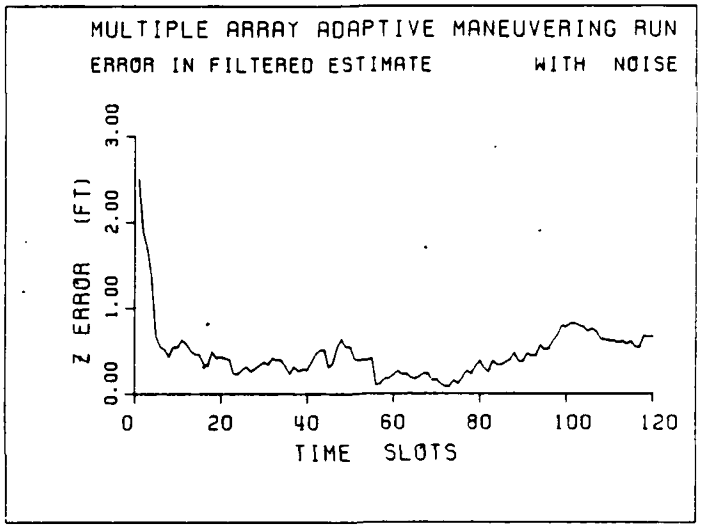


Figure 5.7 Error in Filtered Estimate of Position in Z of the Torpedo During a Maneuvering Run through Multiple Array

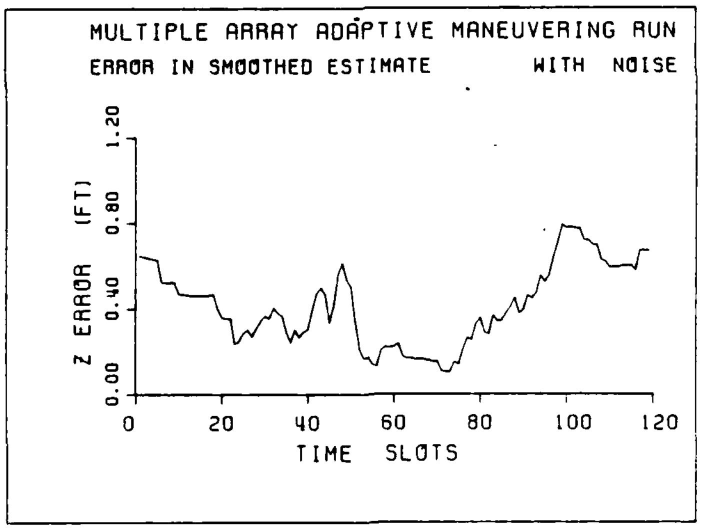


Figure 5.8 Error in Smoothed Estimate of Position in Z of the Torpedo During a Maneuvering Run through Multiple Array

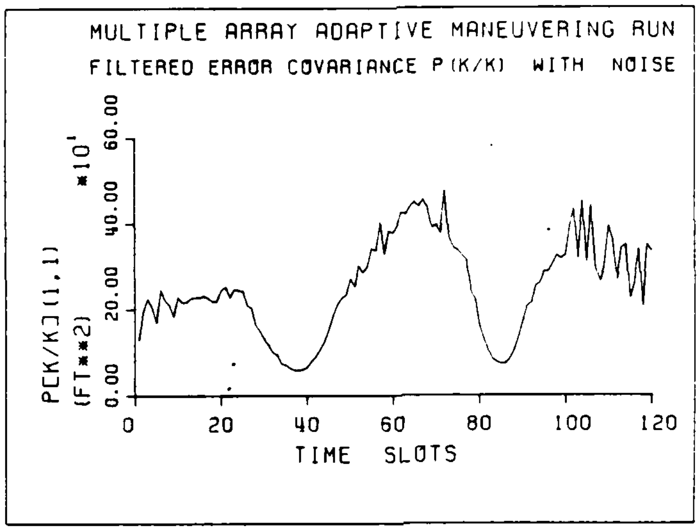


Figure 5.9 Variance of Filtered Position Error in X of the Torpedo During a Maneuvering Run through Multiple Array

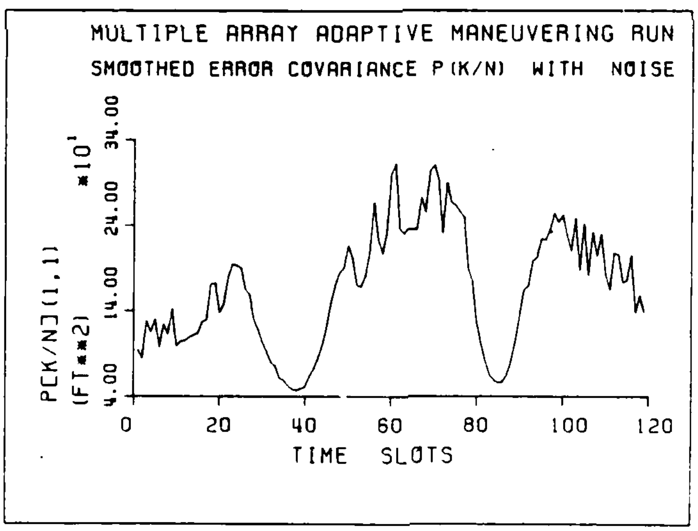


Figure 5.10 Variance of Smoothed Position Error in X of the Torpedo During a Maneuvering Run through Multiple Array

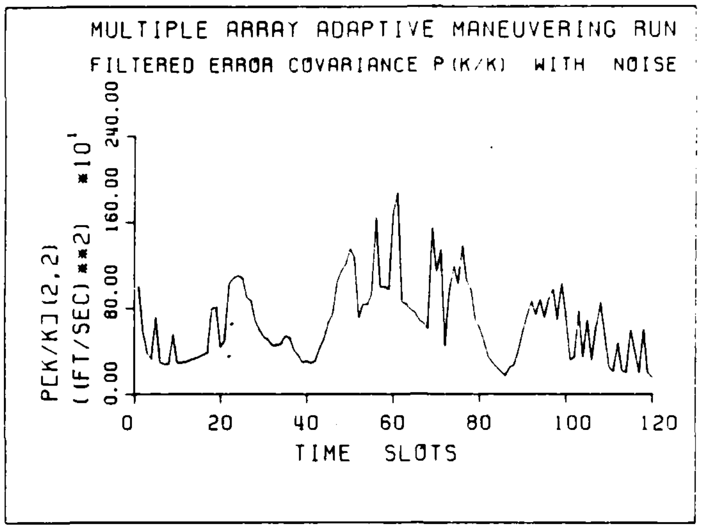


Figure 5.11 Variance of Filtered Velocity Error in X of the Torpedo During a Maneuvering Run through Multiple Array

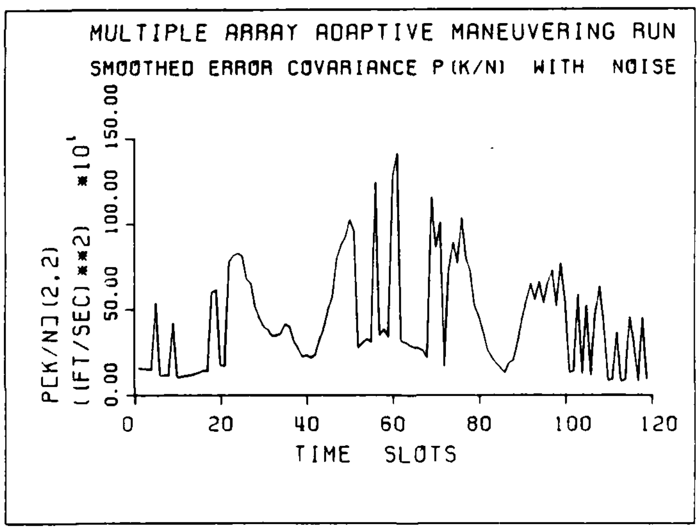


Figure 5.12 Variance of Smoothed Velocity Error in X of the Torpedo During a Maneuvering Run through Multiple Array

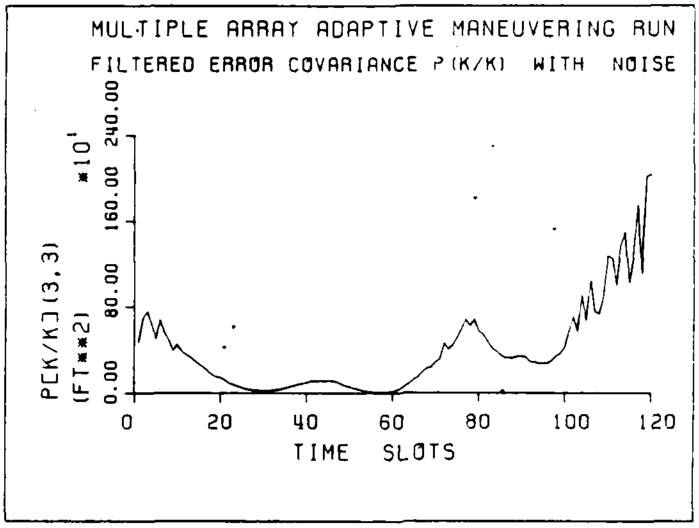


Figure 5.13 Variance of Filtered Position Error in Y of the Torpedo During a Maneuvering Run through Multiple Array

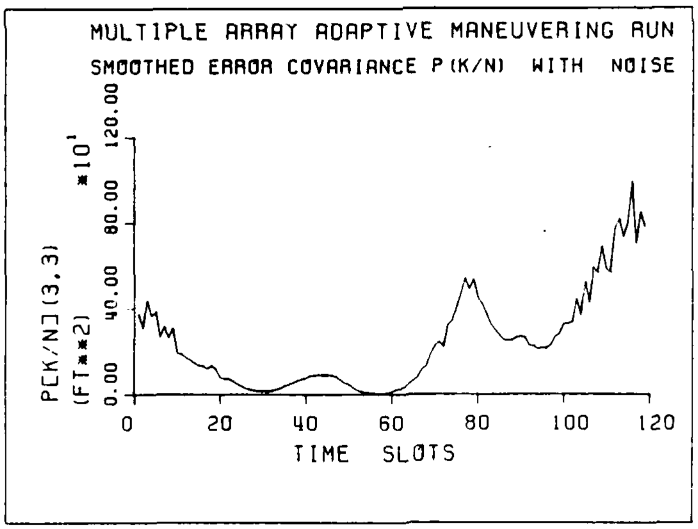


Figure 5.14 Variance of Smoothed Position Error in Y of the Torpedo During a Maneuvering Run through Multiple Array

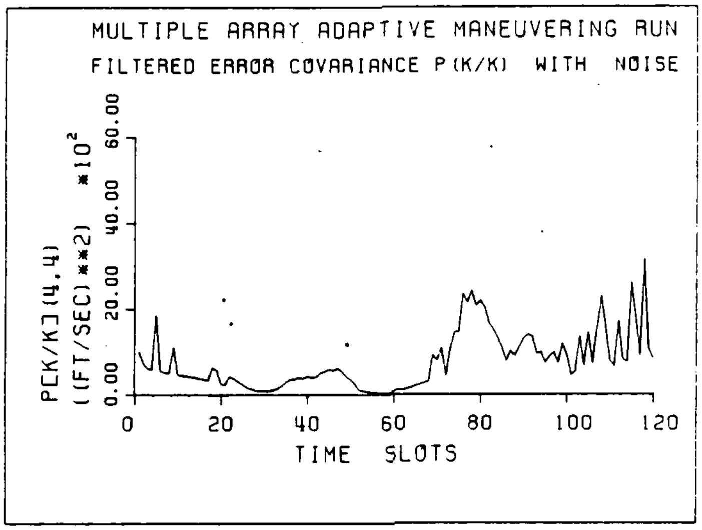


Figure 5.15 Variance of Filtered Velocity Error in Y of the Torpedo During a Maneuvering Run through Multiple Array

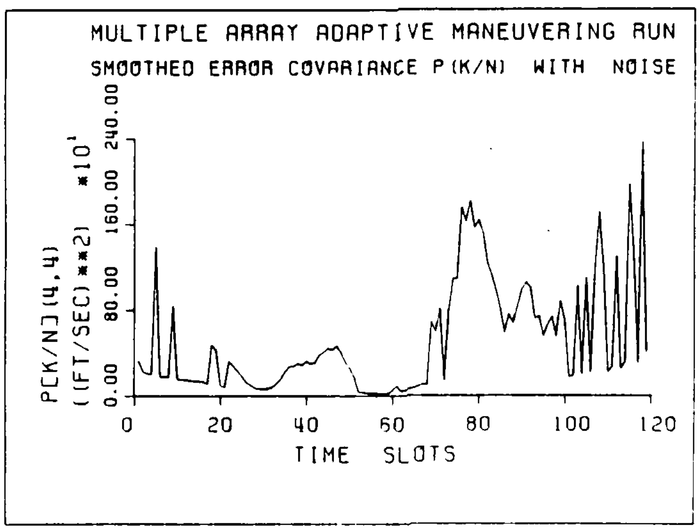


Figure 5.16 Variance of Smoothed Velocity Error in Y of the Torpedo During a Maneuvering Run through Multiple Array

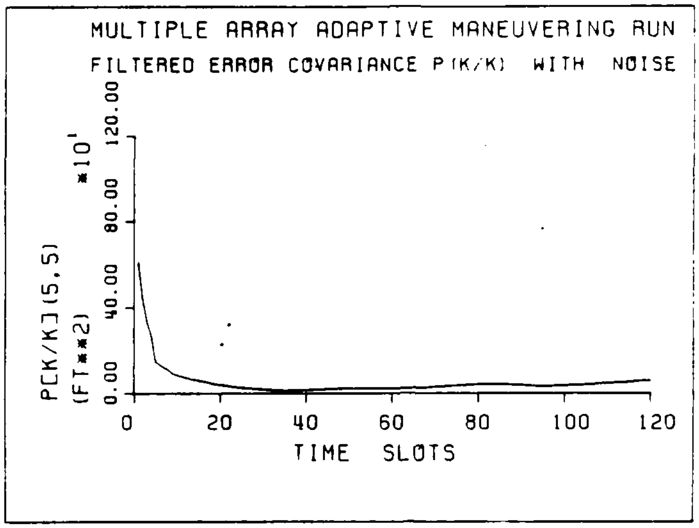


Figure 5.17 Variance of Filtered Position Error in Z of the Torpedo During a Maneuvering Run through Multiple Array

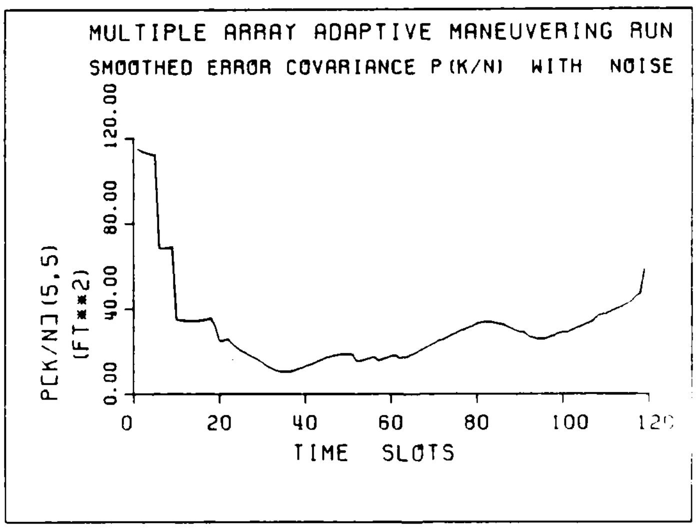


Figure 5.18 Variance of Smoothed Position Error in Z of the Torpedo During a Maneuvering Run through Multiple Array

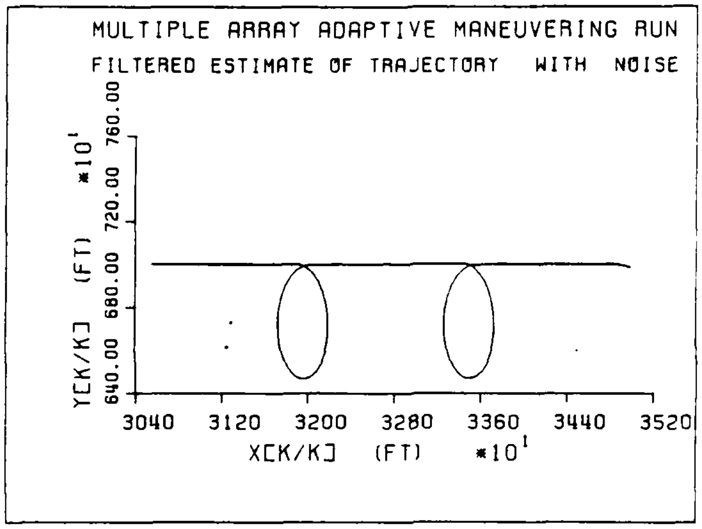


Figure 5.19 Filtered Estimate of Trajectory of the Torpedo During a Maneuvering Run through Multiple Array

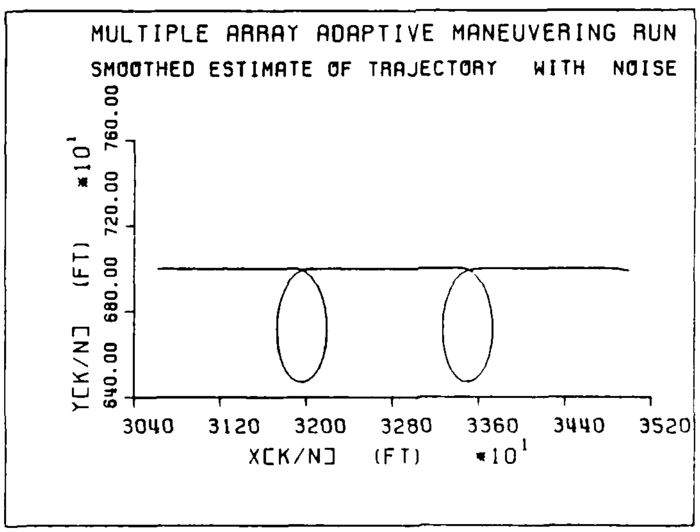


Figure 5.20 Smoothed Estimate of Trajectory of the Torpedo During a Maneuvering Run through Multiple Array

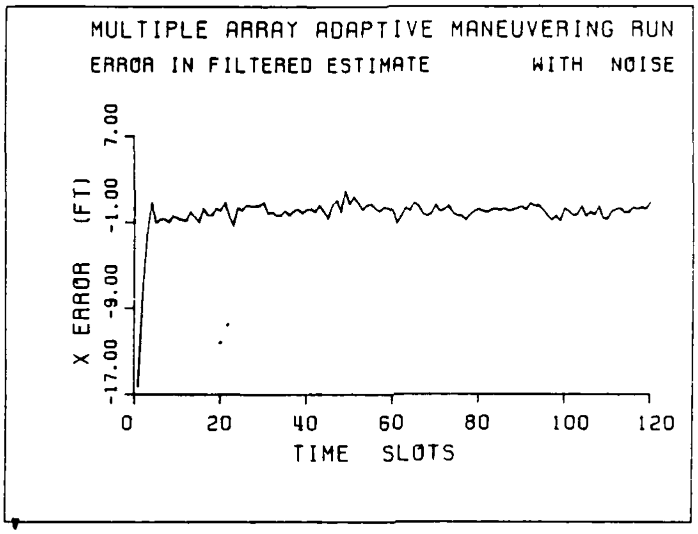


Figure 5.21 Error in Filtered Estimate of Position in X of the Torpedo During a Maneuvering Run through Multiple Array

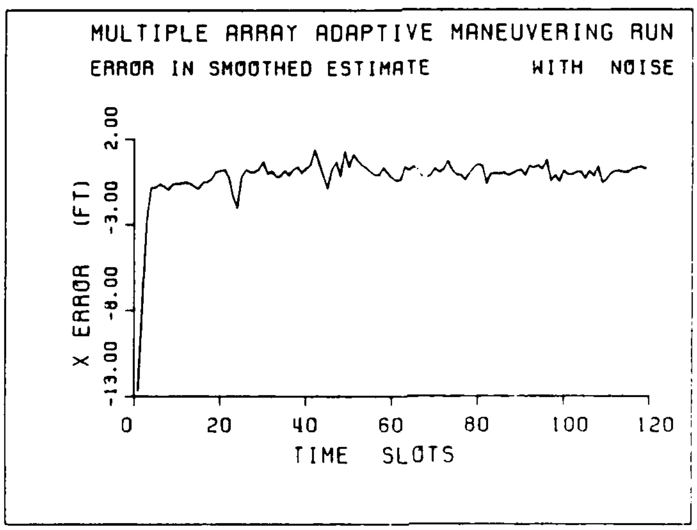


Figure 5.22 Error in Smoothed Estimate of Position in X of the Torpedo During a Maneuvering Run through Multiple Array

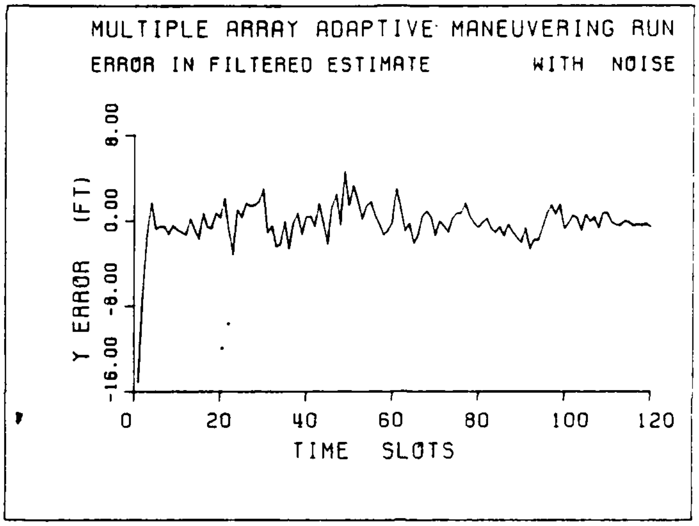


Figure 5.23 Error in Filtered Estimate of Position in Y of the Torpedo During a Maneuvering Run through Multiple Array

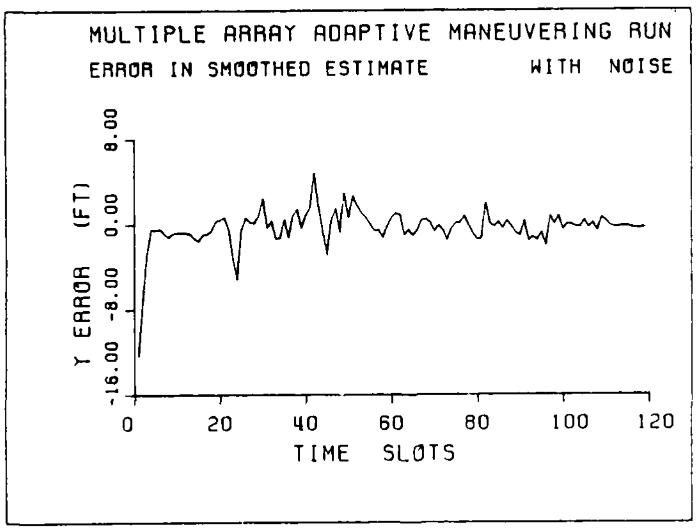


Figure 5.24 Error in Smoothed Estimate of Position in Y of the Torpedo During a Maneuvering Run through Multiple Array

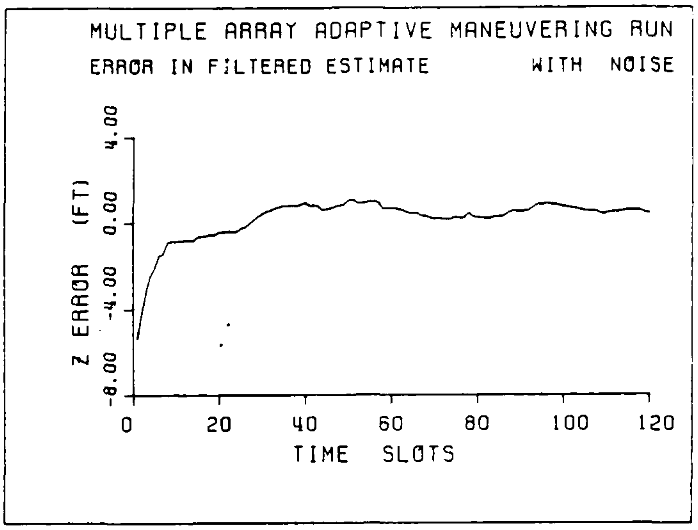


Figure 5.25 Error in Filtered Estimate of Position in Z of the Torpedo During a Maneuvering Run through Multiple Array

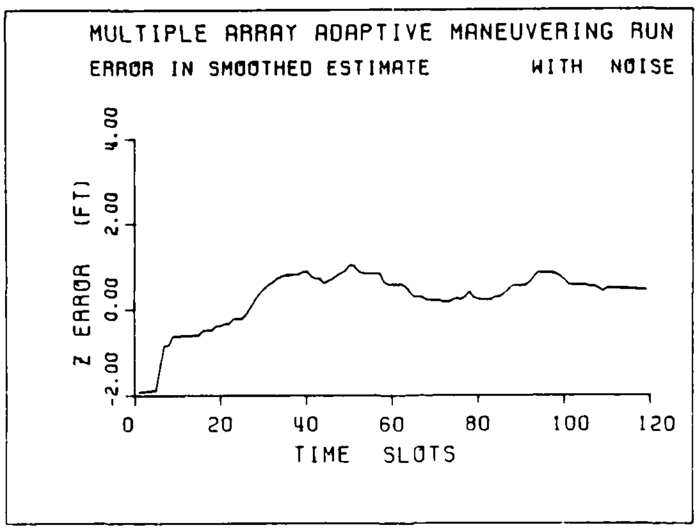


Figure 5.26 Error in Smoothed Estimate of Position in Z of the Torpedo During a Maneuvering Run through Multiple Array

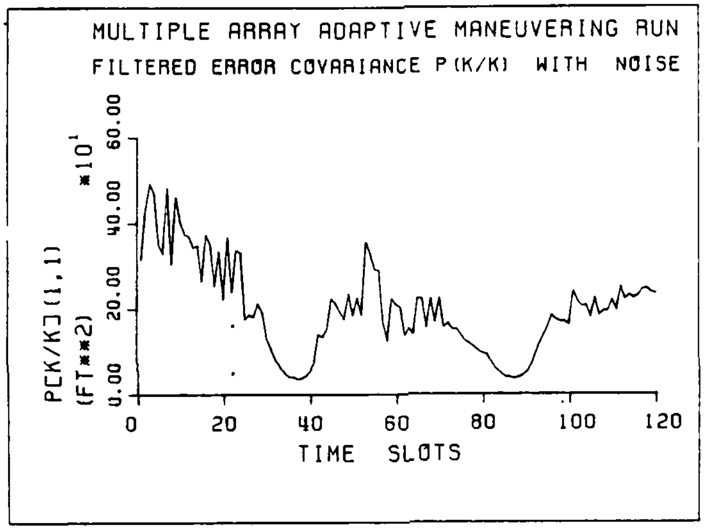


Figure 5.27 Variance of Filtered Position Error in X of the Torpedo During a Maneuvering Run through Multiple Array

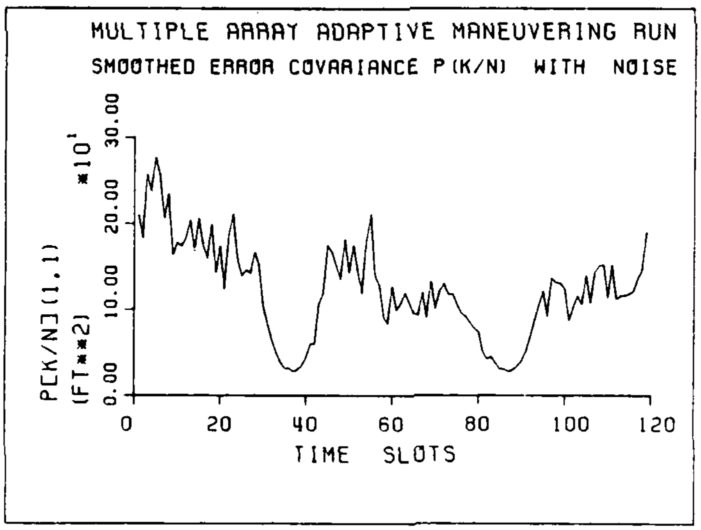


Figure 5.28 Variance of Smoothed Position Error in X of the Torpedo During a Maneuvering Run through Multiple Array

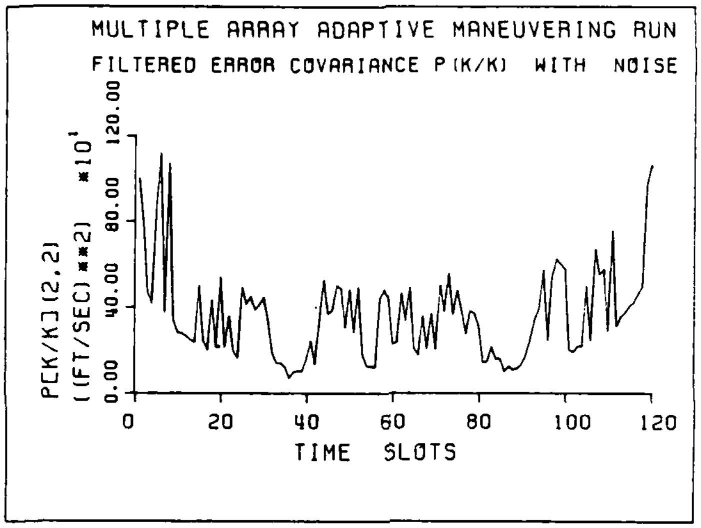


Figure 5.29 Variance of Filtered Velocity Error in X of the Torpedo During a Maneuvering Run through Multiple Array

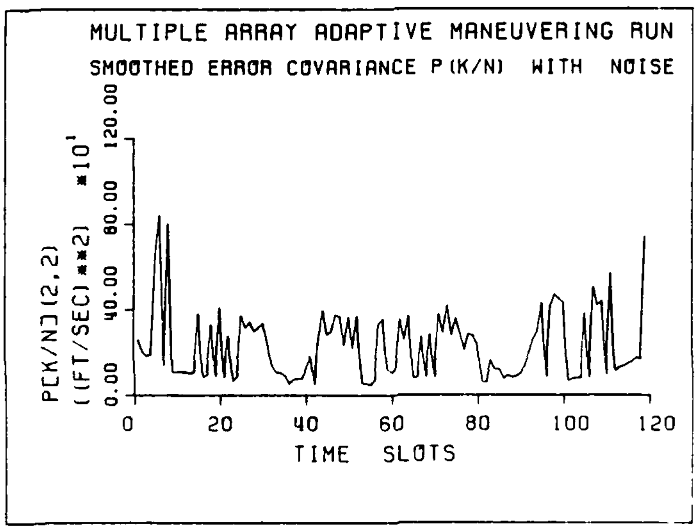


Figure 5.30 Variance of Smoothed Velocity Error in X of the Torpedo During a Maneuvering Run through Multiple Array

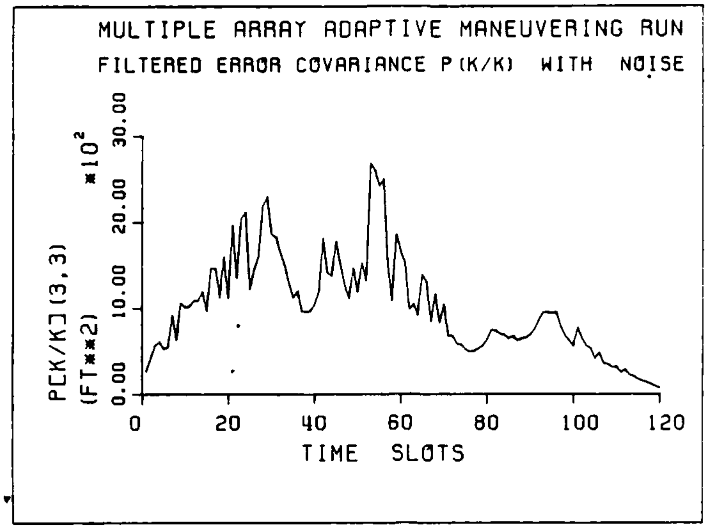


Figure 5.31 Variance of Filtered Position Error in Y of the Torpedo During a Maneuvering Run through Multiple Array

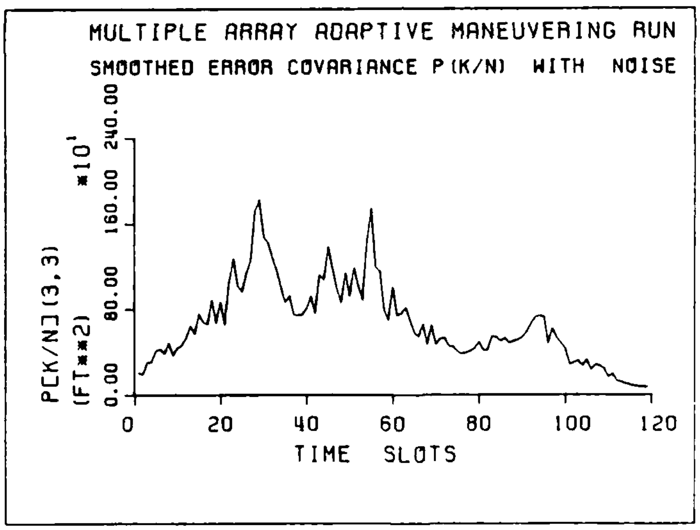


Figure 5.32 Variance of Smoothed Position Error in Y of the Torpedo During a Maneuvering Run through Multiple Array

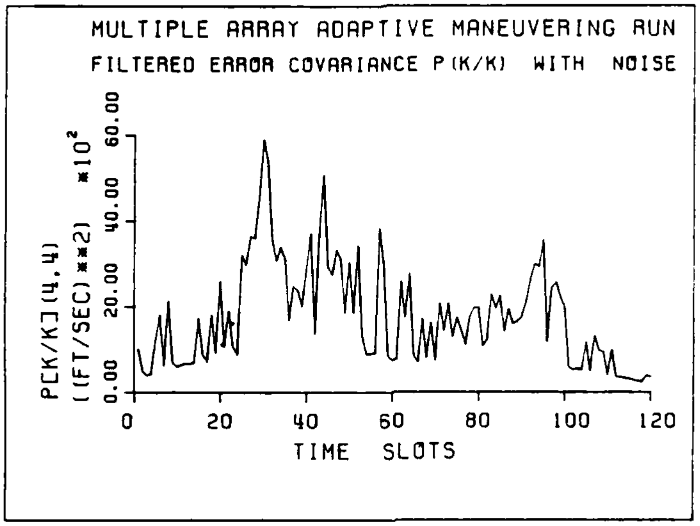


Figure 5.33 Variance of Filtered Velocity Error in Y of the Torpedo During a Maneuvering Run through Multiple Array

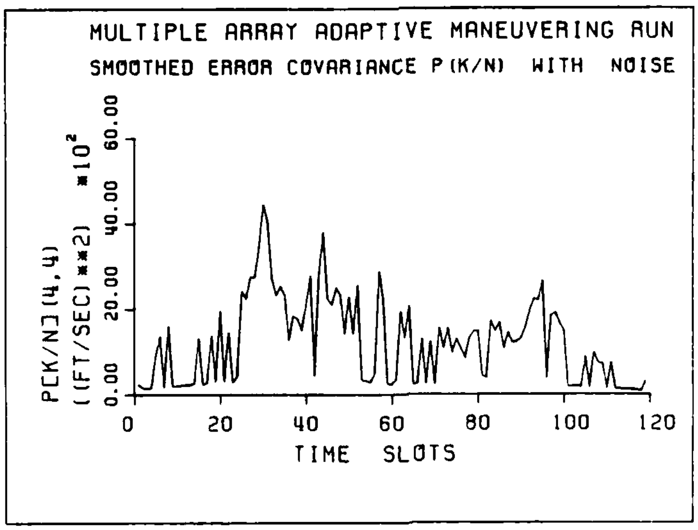


Figure 5.34 Variance of Smoothed Velocity Error in Y of the Torpedo During a Maneuvering Run through Multiple Array

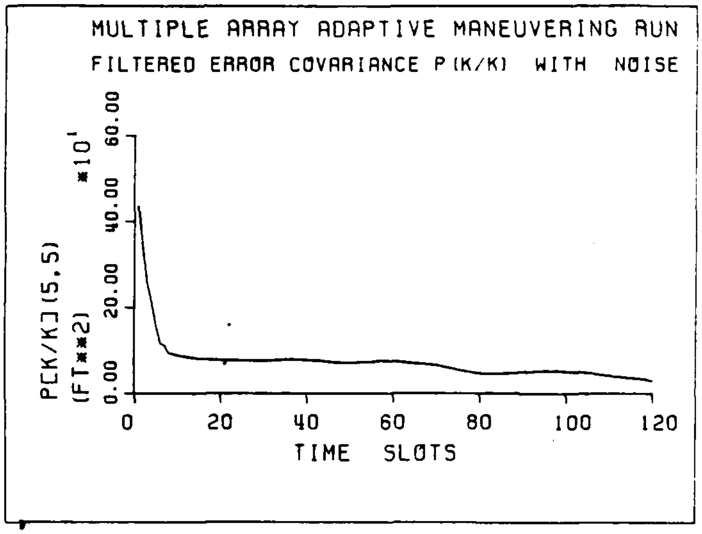


Figure 5.35 Variance of Filtered Position Error in Z of the Torpedo During a Maneuvering Run through Multiple Array

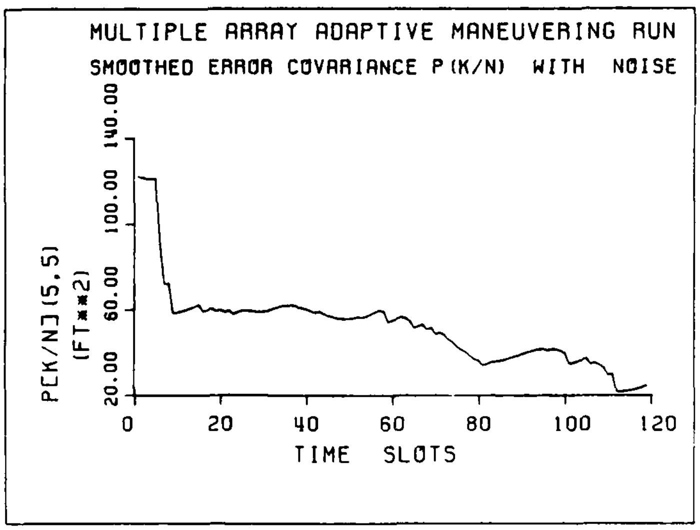


Figure 5.36 Variance of Smoothed Position Error in Z of the Torpedo During a Maneuvering Run through Multiple Array

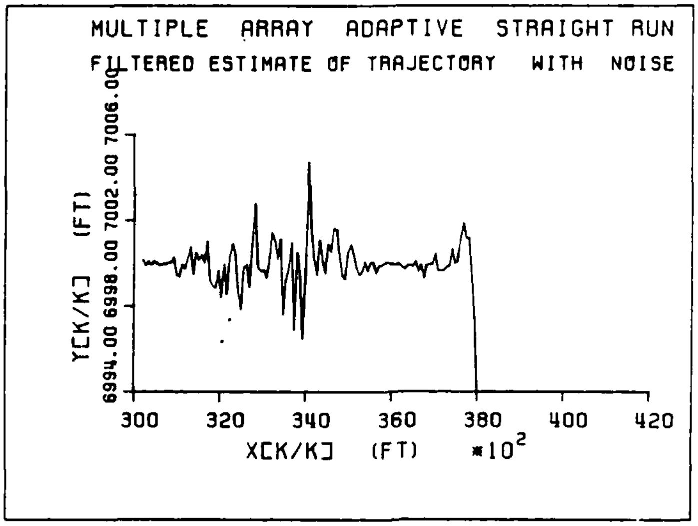


Figure 5.37 Filtered Estimate of Trajectory of the Torpedo During a Straight Run through Multiple Array

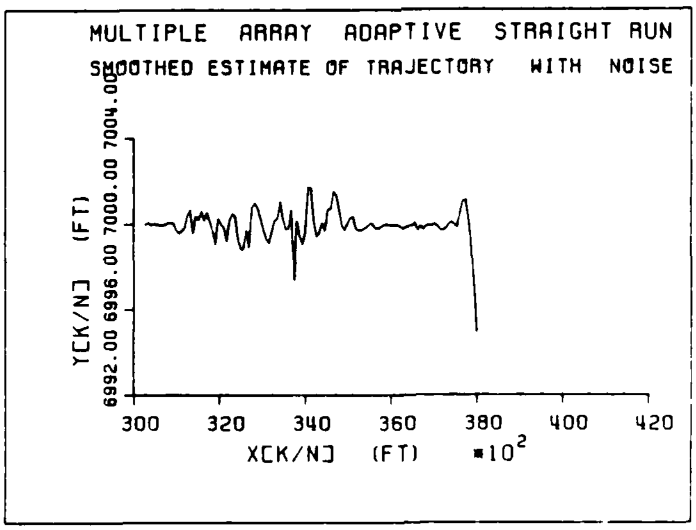


Figure 5.38 Smoothed Estimate of Trajectory of the Torpedo During a Straight Run through Multiple Array

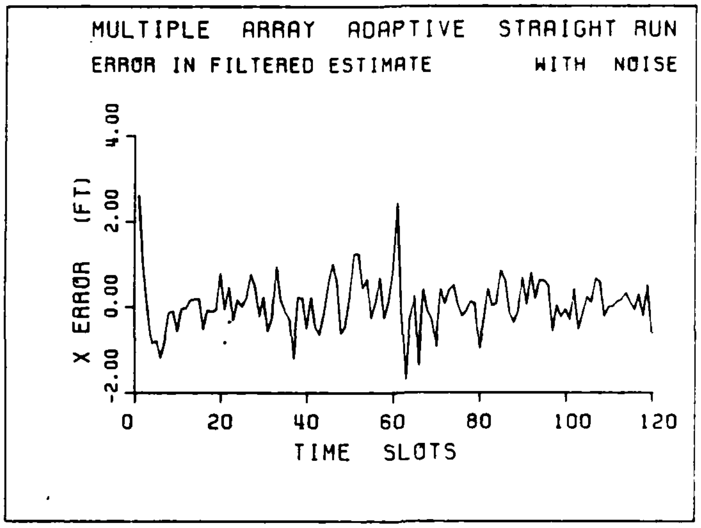


Figure 5.39 Error in Filtered Estimate of Position in X of the Torpedo During a Straight Run through Multiple Array

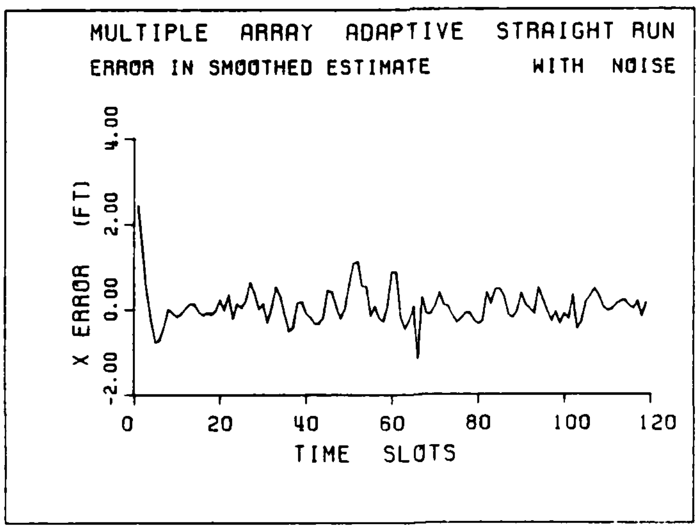


Figure 5.40 Error in Smoothed Estimate of Position in X of the Torpedo During a Straight Run through Multiple Array

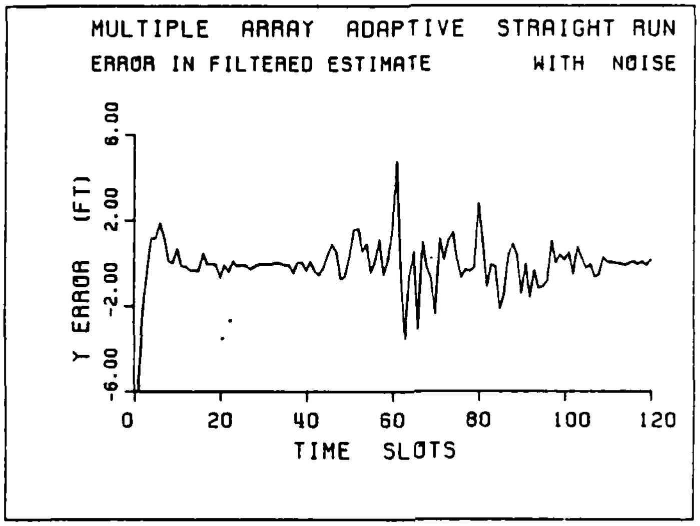


Figure 5.41 Error in Filtered Estimate of Position in Y of the Torpedo During a Straight Run through Multiple Array

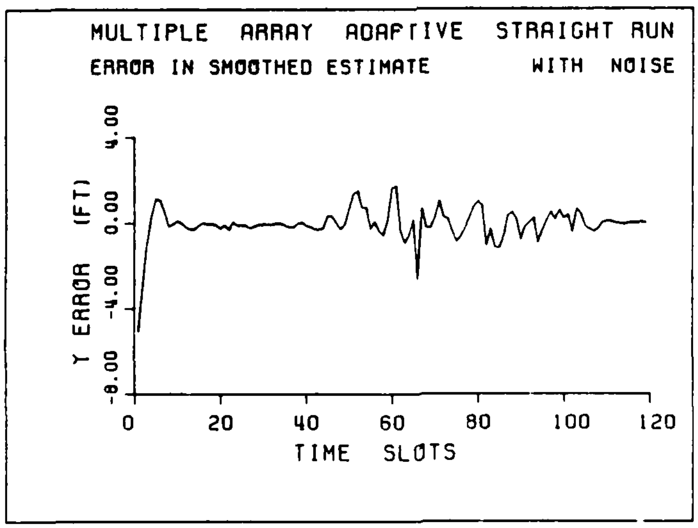


Figure 5.42 Error in Smoothed Estimate of Position in Y of the Torpedo During a Straight Run through Multiple Array

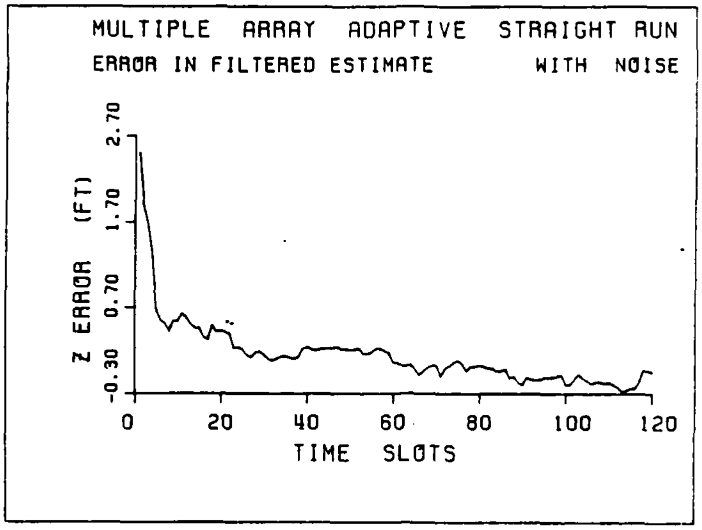


Figure 5.43 Error in Filtered Estimate of Position in Z of the Torpedo During a Straight Run through Multiple Array

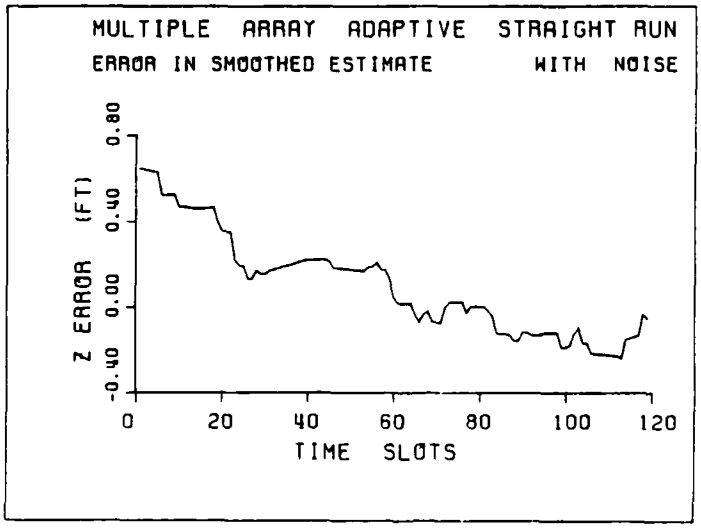


Figure 5.44 Error in Smoothed Estimate of Position in Z of the Torpedo During a Straight Run through Multiple Array

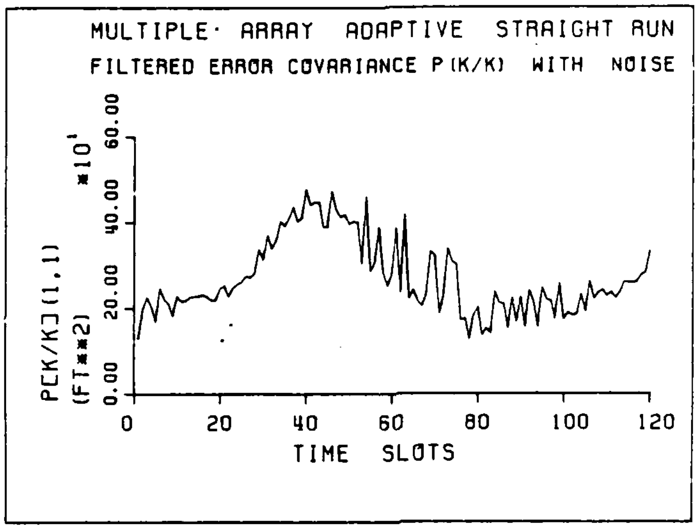


Figure 5.45 Variance of Filtered Position Error in X of the Torpedo During a Straight Run through Multiple Array

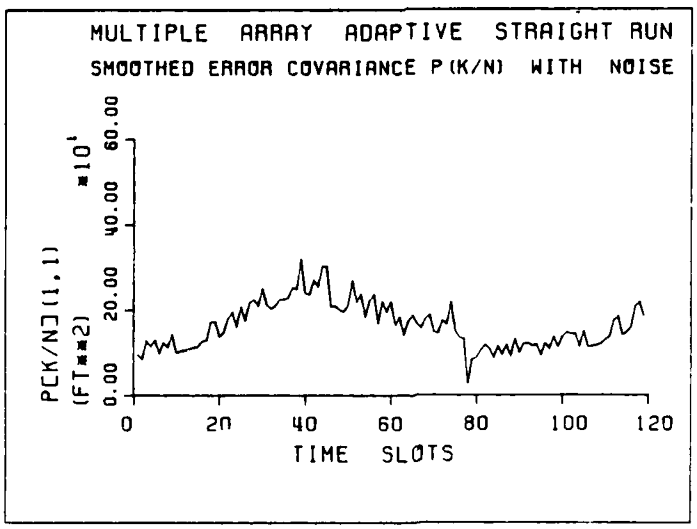


Figure 5.46 Variance of Smoothed Position Error in X of the Torpedo During a Straight Run through Multiple Array

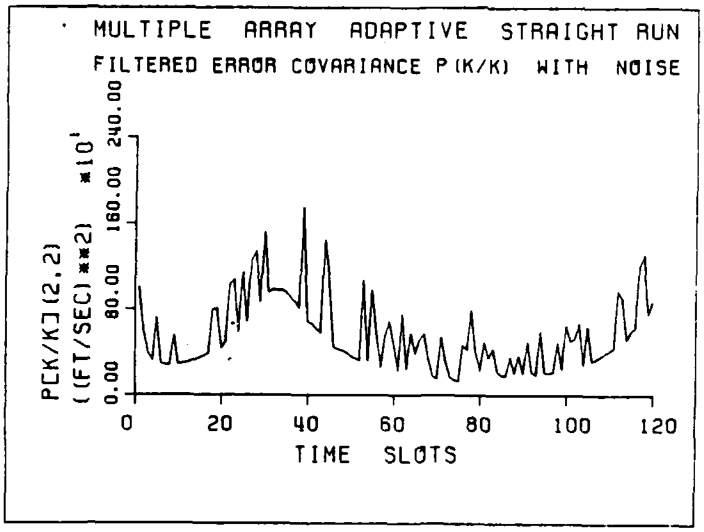


Figure 5.47 Variance of Filtered Velocity Error in X of the Torpedo During a Straight Run through Multiple Array

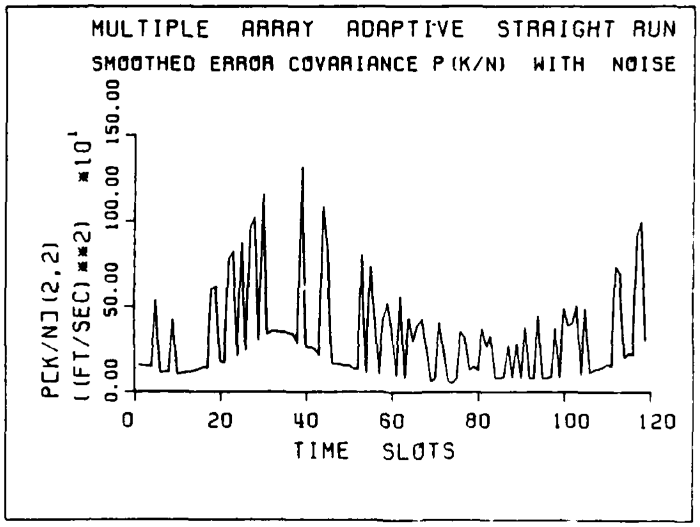


Figure 5.48 Variance of Smoothed Velocity Error in X of the Torpedo During a Straight Run through Multiple Array

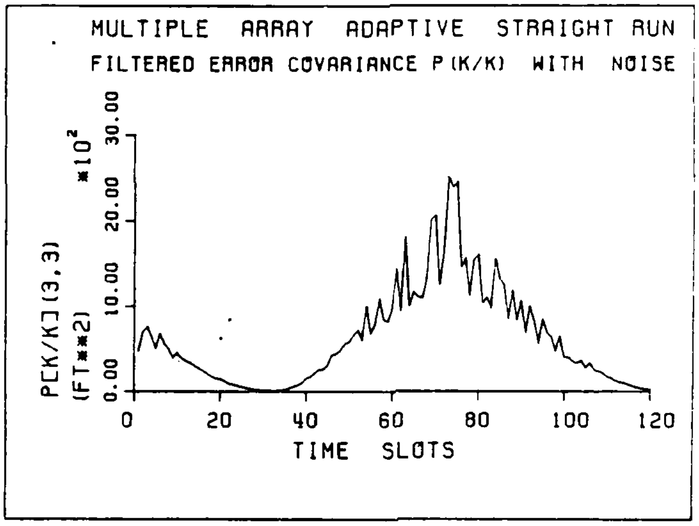


Figure 5.49 Variance of Filtered Position Error in Y of the Torpedo During a Straight Run through Multiple Array

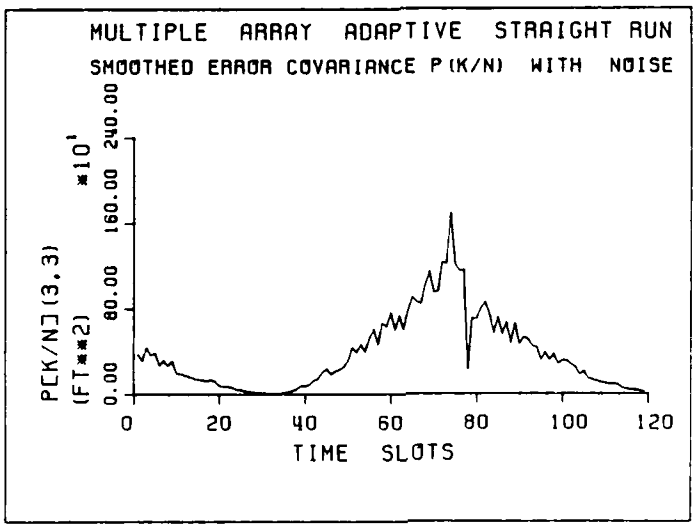


Figure 5.50 Variance of Smoothed Position Error in Y of the Torpedo During a Straight Run through Multiple Array

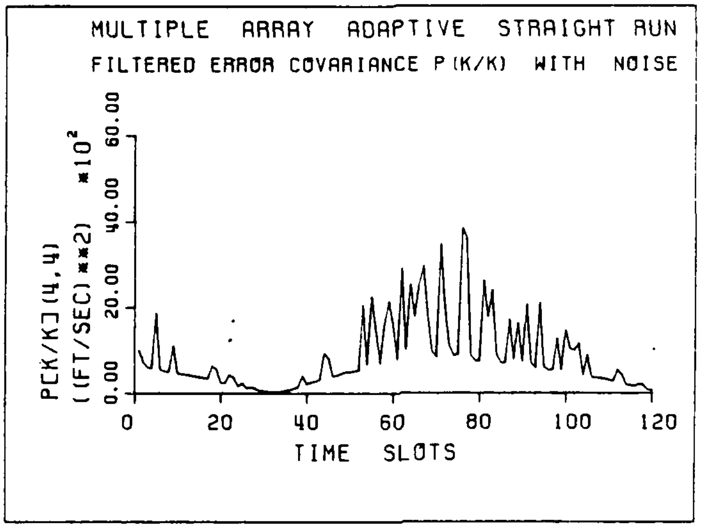


Figure 5.51 Variance of Filtered Velocity Error in Y of the Torpedo During a Straight Run through Multiple Array

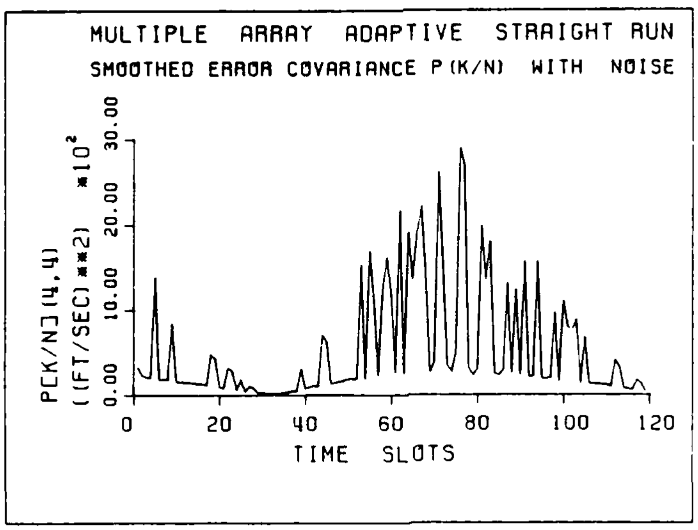


Figure 5.52 Variance of Smoothed Velocity Error in Y of the Torpedo During a Straight Run through Multiple Array

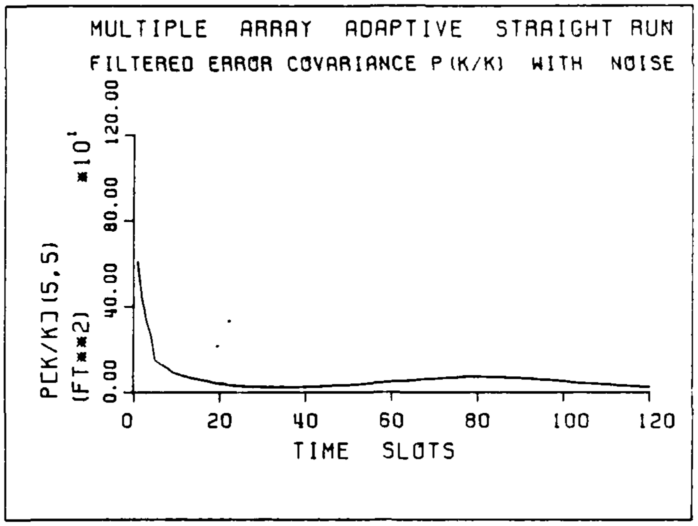


Figure 5.53 Variance of Filtered Position Error in Z of the Torpedo During a Straight Run through Multiple Array

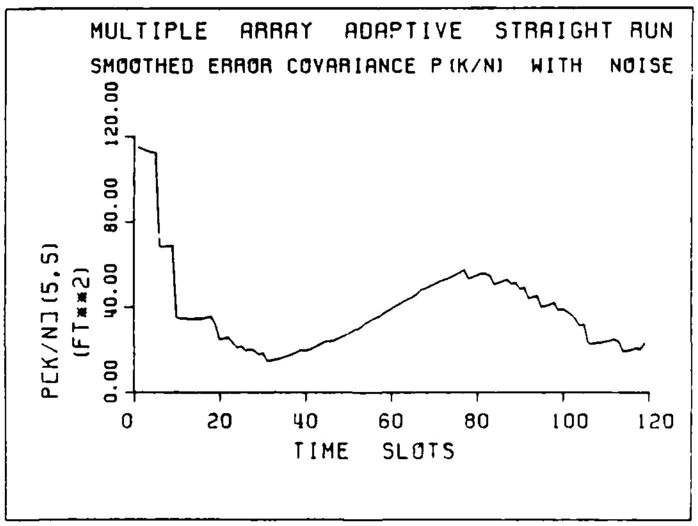


Figure 5.54 Variance of Smoothed Position Error in Z of the Torpedo During a Straight Run through Multiple Array

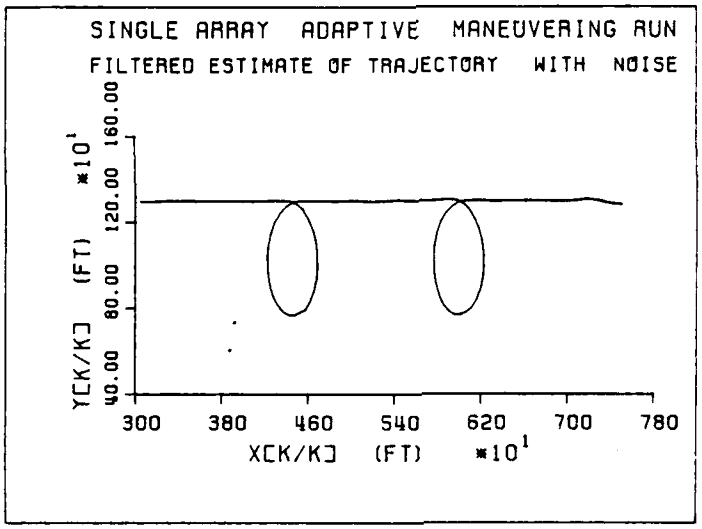


Figure 5.55 Filtered Estimate of Trajectory of the Torpedo During a Maneuvering Run through Single Array

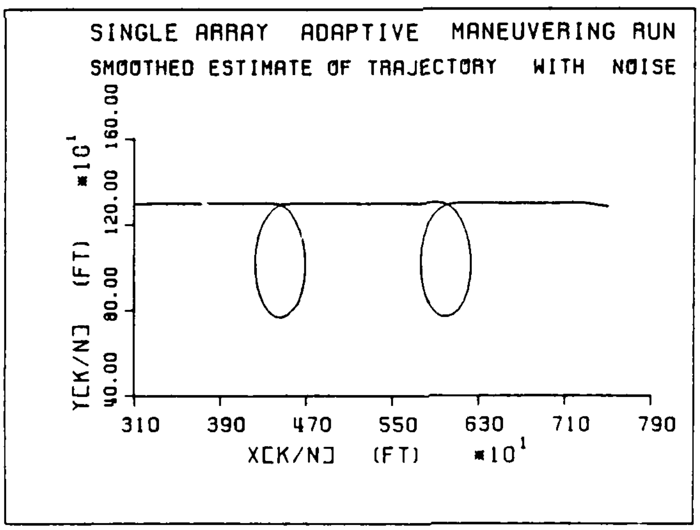


Figure 5.56 Smoothed Estimate of Trajectory of the Torpedo During a Maneuvering Run through Single Array

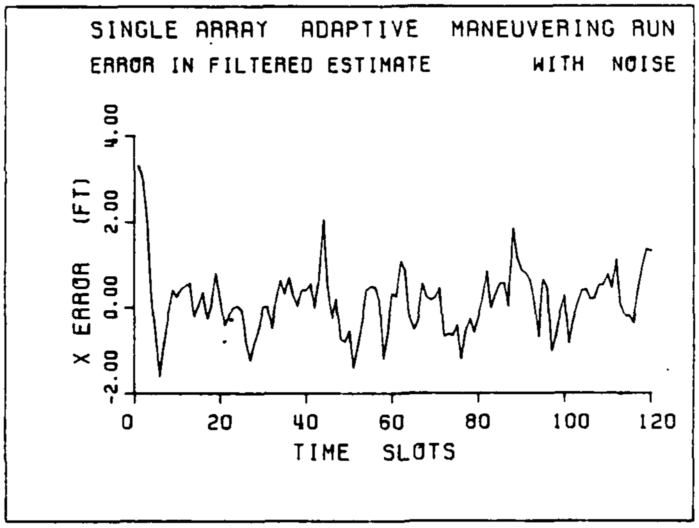


Figure 5.57 Error in Filtered Estimate of Position in X of the Torpedo During a Maneuvering Run through Single Array

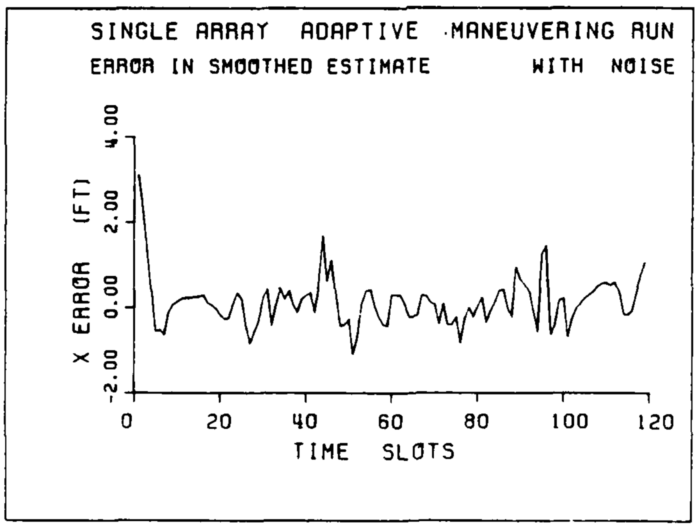


Figure 5.58 Error in Smoothed Estimate of Position in X of the Torpedo During a Maneuvering Run through Single Array

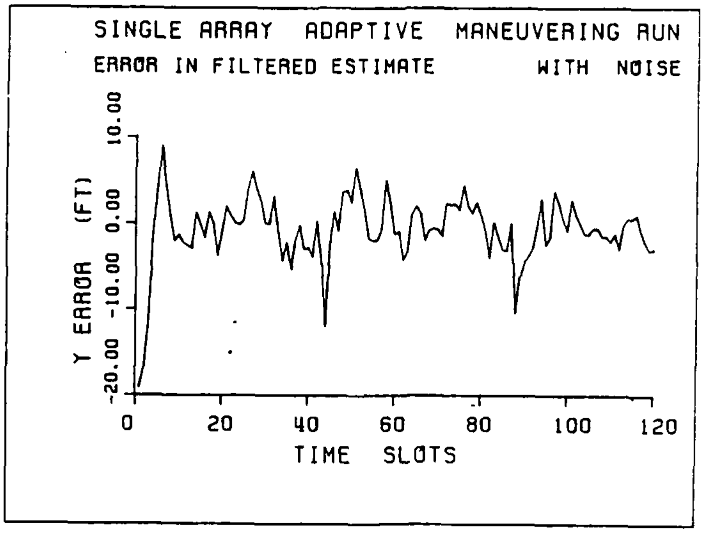


Figure 5.59 Error in Filtered Estimate of Position in Y of the Torpedo During a Maneuvering Run through Single Array

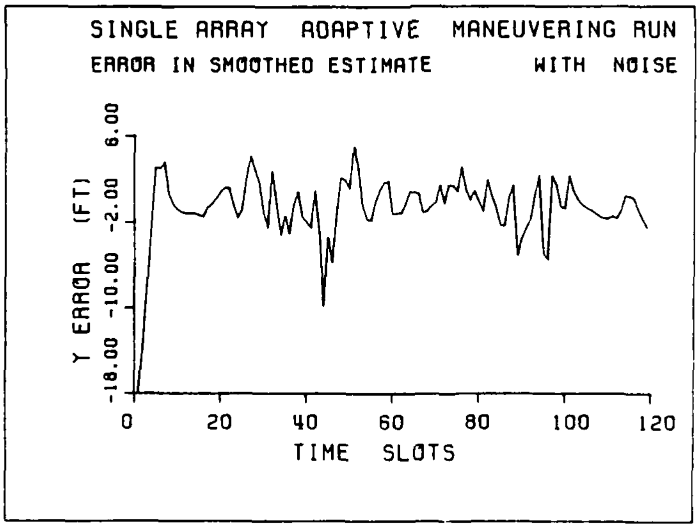


Figure 5.60 Error in Smoothed Estimate of Position in Y of the Torpedo During a Maneuvering Run through Single Array

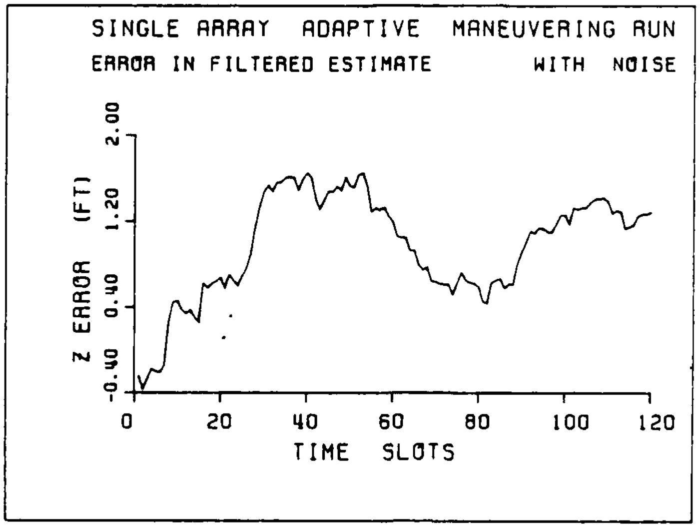


Figure 5.61 Error in Filtered Estimate of Position in Z of the Torpedo During a Maneuvering Run through Single Array

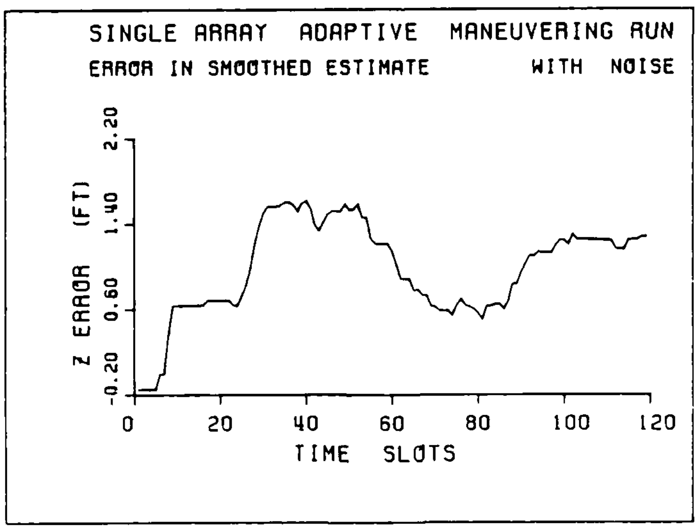


Figure 5.62 Error in Smoothed Estimate of Position in Z of the Torpedo During a Maneuvering Run through Single Array

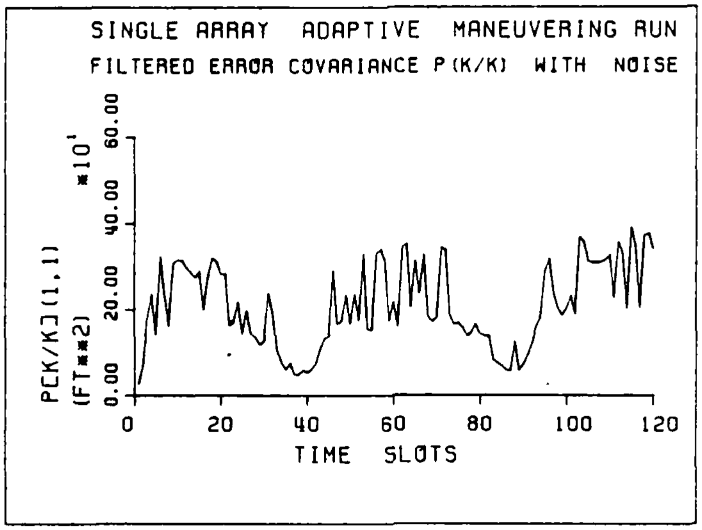


Figure 5.63 Variance of Filtered Position Error in X of the Torpedo During a Maneuvering Run through Single Array

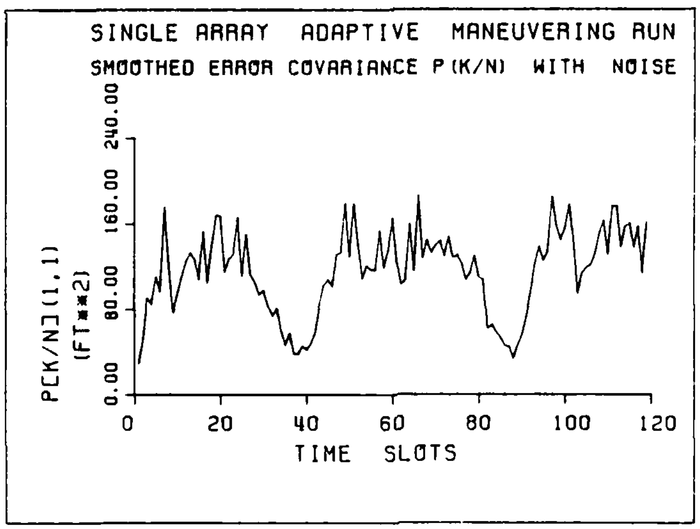


Figure 5.64 Variance of Smoothed Position Error in X of the Torpedo During a Maneuvering Run through Single Array

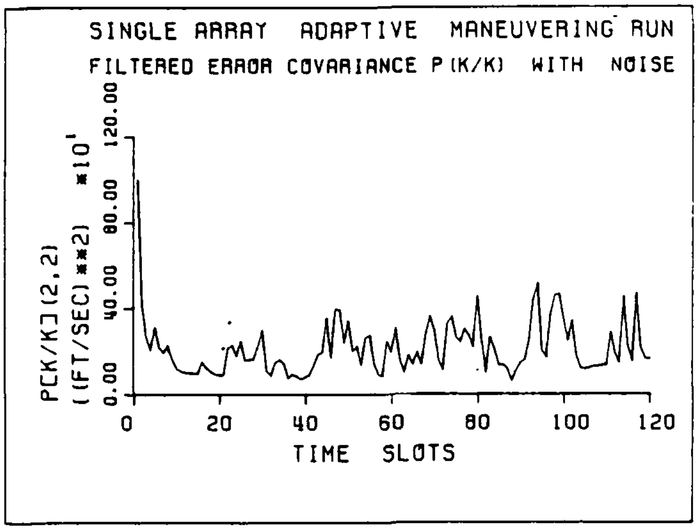


Figure 5.65 Variance of Filtered Velocity Error in X of the Torpedo During a Maneuvering Run through Sinlge Array

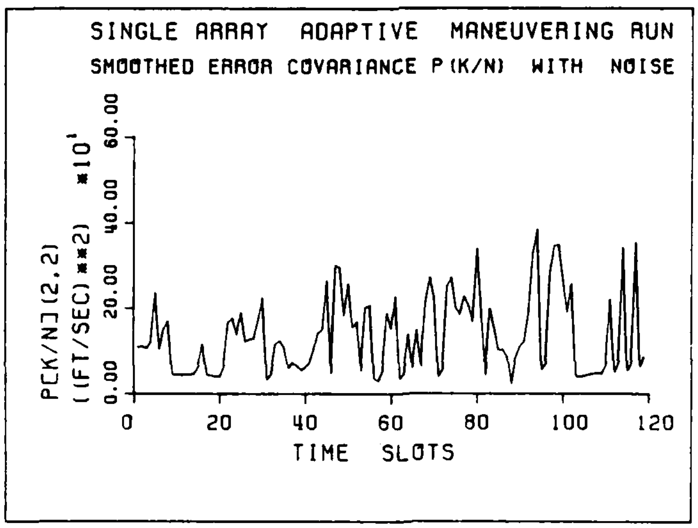


Figure 5.66 Variance of Smoothed Velocity Error in X of the Torpedo During a Maneuvering Run through Single Array

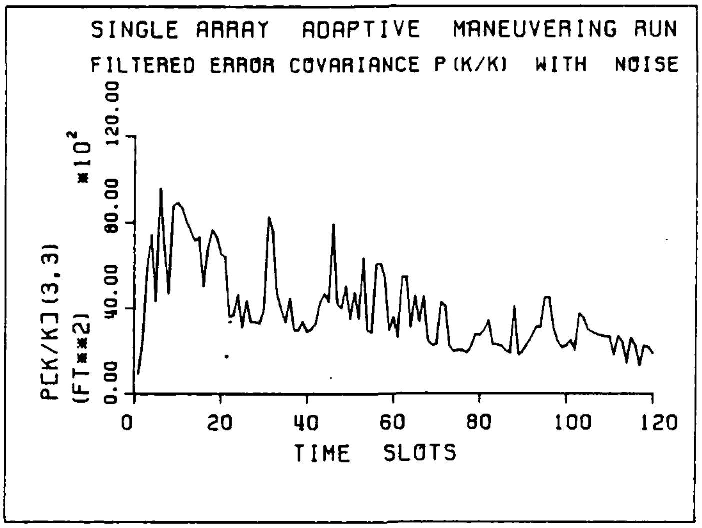


Figure 5.67 Variance of Filtered Position Error in Y of the Torpedo During a Maneuvering Run through Single Array

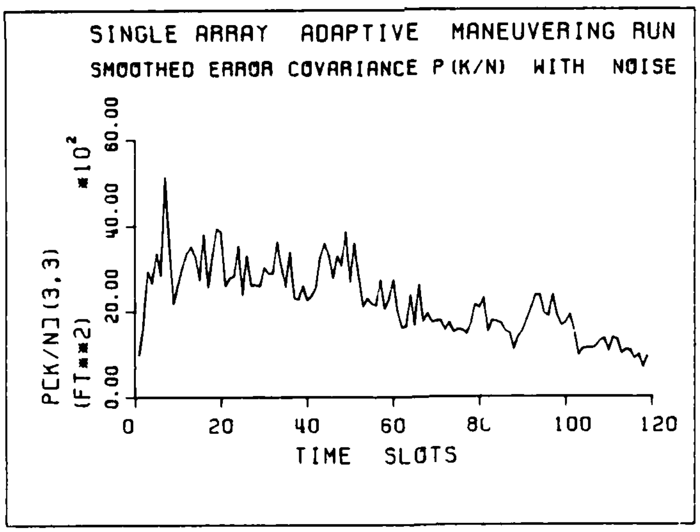


Figure 5.68 Variance of Smoothed Position Error in Y of the Torpedo During a Maneuvering Run through Single Array

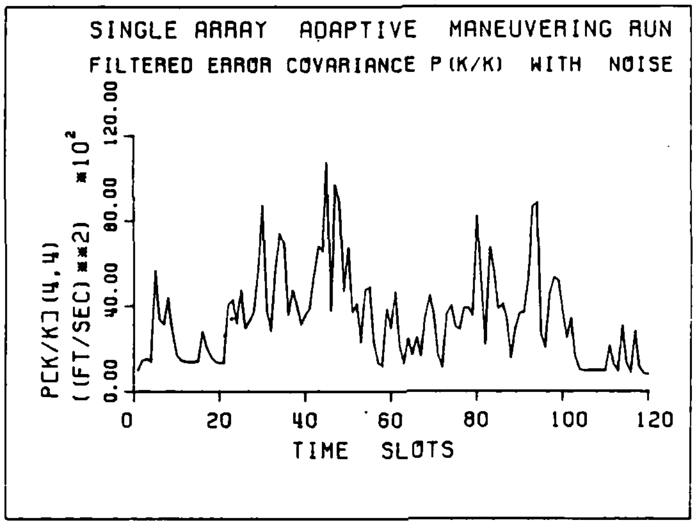


Figure 5.69 Variance of Filtered Velocity Error in Y of the Torpedo During a Maneuvering Run through Single Array

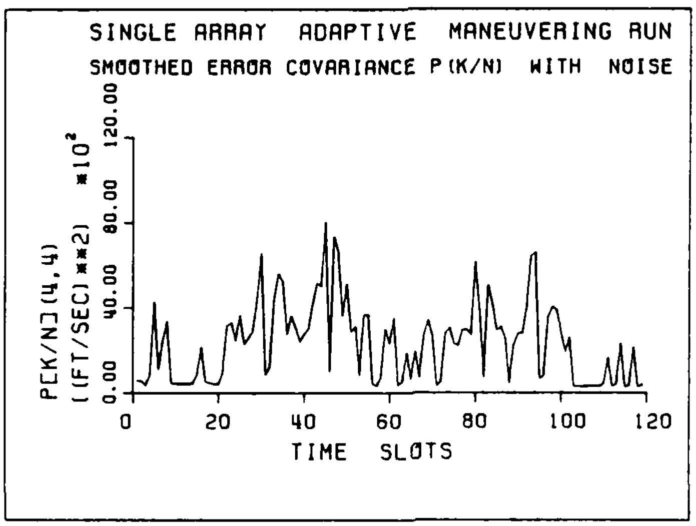


Figure 5.70 Variance of Smoothed Velocity Error in Y of the Torpedo During a Maneuvering Run through Single Array

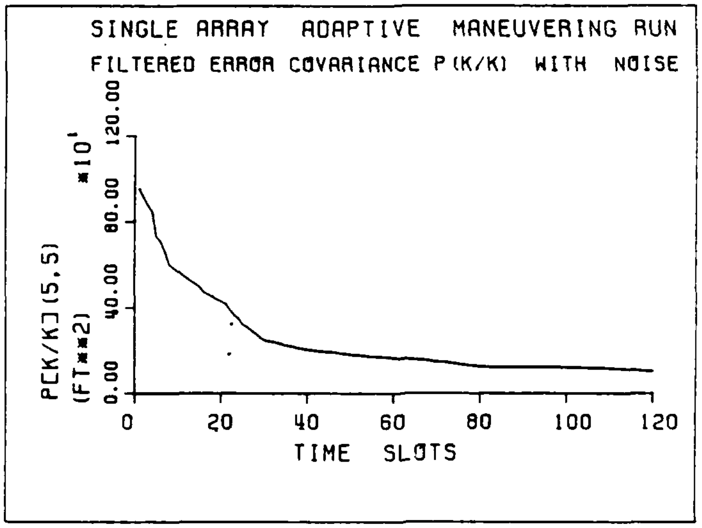


Figure 5.71 Variance of Filtered Position Error in Z of the Torpedo During a Maneuvering Run through Single Array

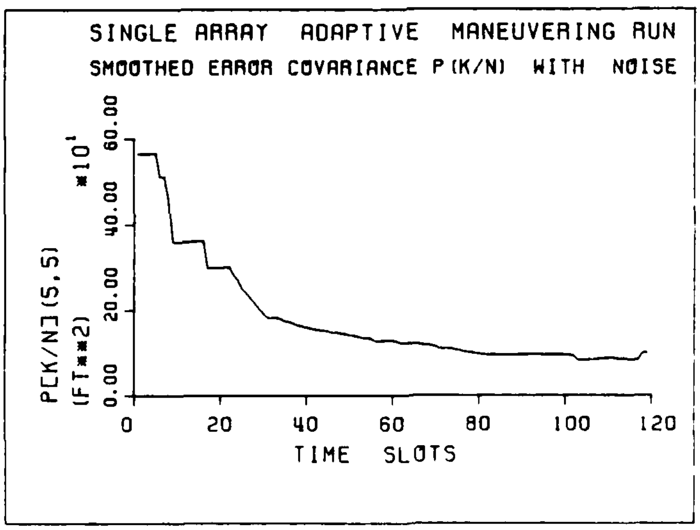


Figure 5.72 Variance of Smoothed Position Error in Z of the Torpedo During a Maneuvering Run through Single Array

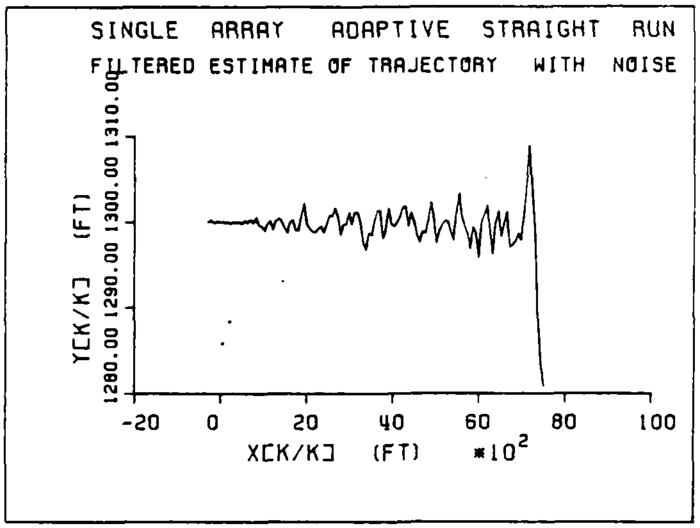


Figure 5.73 Filtered Estimate of Trajectory of the Torpedo During a Straight Run through Single Array

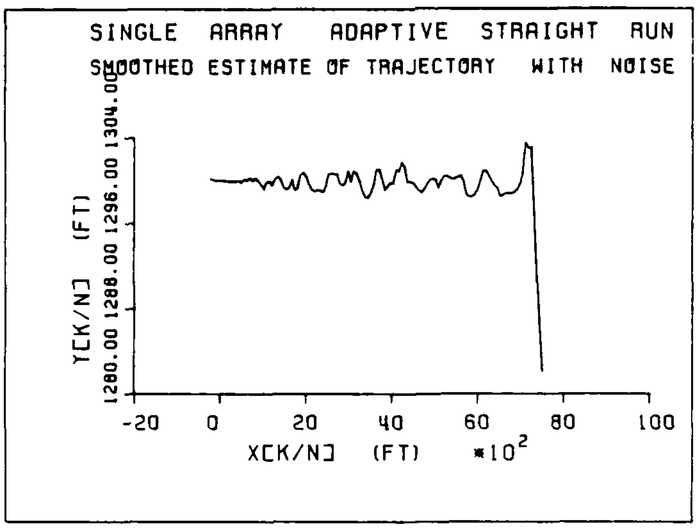


Figure 5.74 Smoothed Estimate of Trajectory of the Torpedo During a Straight Run through Single Array

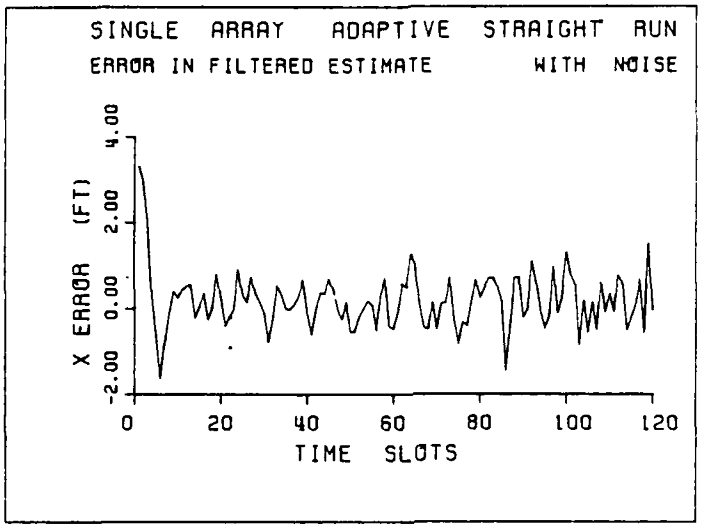


Figure 5.75 Error in Filtered Estimate of Position in X of the Torpedo During a Straight Run through Single Array

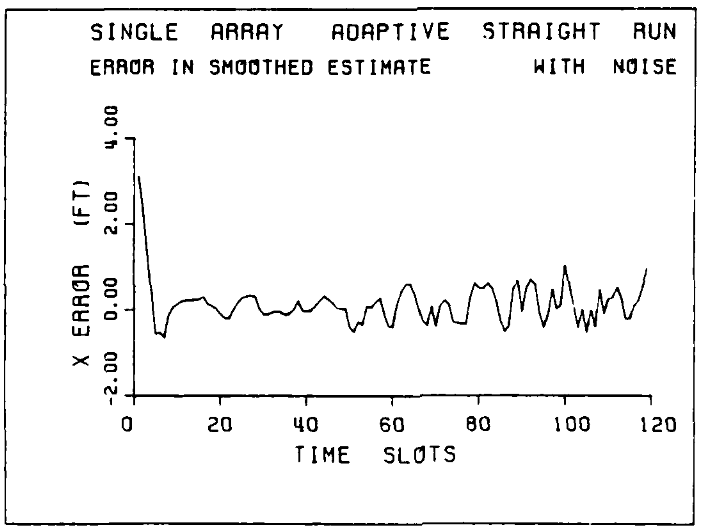


Figure 5.76 Error in Smoothed Estimate of Position in X of the Torpedo During a Straight Run through Single Array

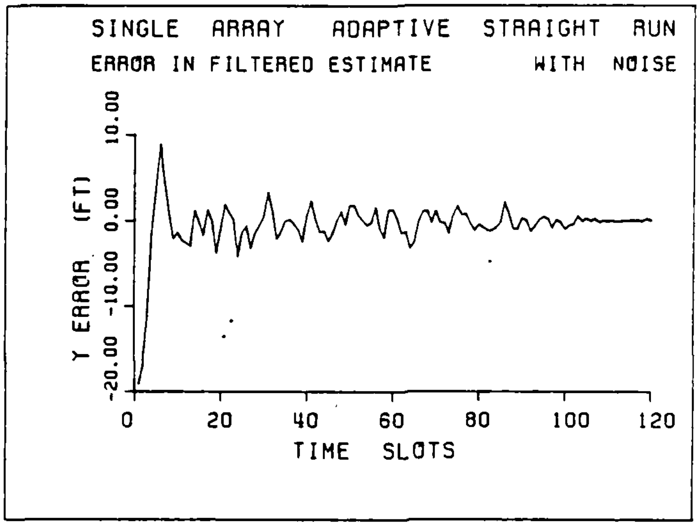


Figure 5.77 Error in Filtered Estimate of Position in Y of the Torpedo During a Straight Run through Single Array

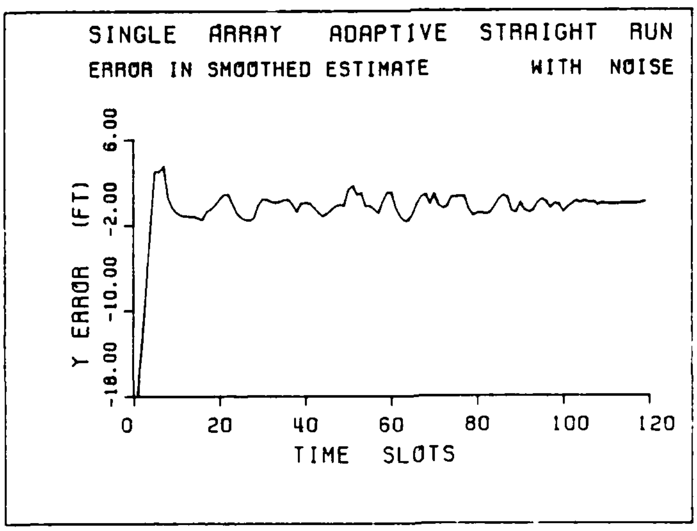


Figure 5.78 Error in Smoothed Estimate of Position in Y of the Torpedo During a Straight Run through Single Array

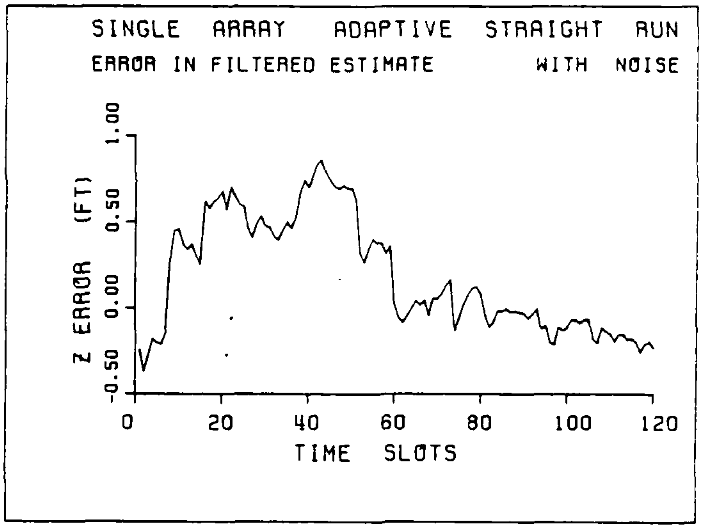


Figure 5.79 Error in Filtered Estimate of Position in Z of the Torpedo During a Straight Run through Single Array

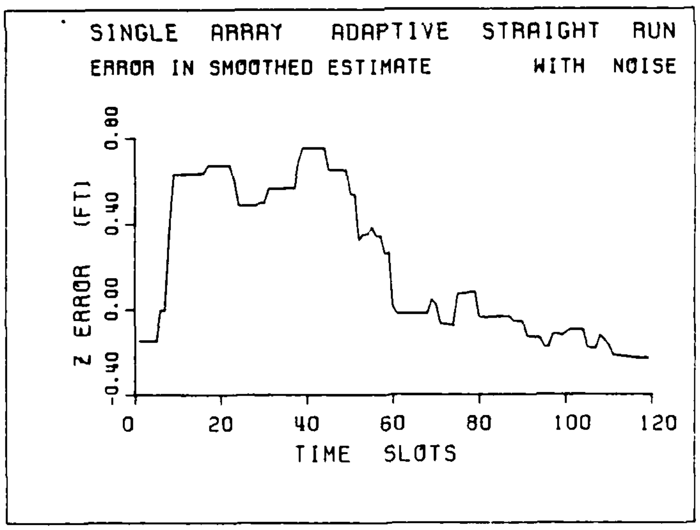


Figure 5.80 Error in Smoothed Estimate of Position in Z of the Torpedo During a Straight Run through Single Array

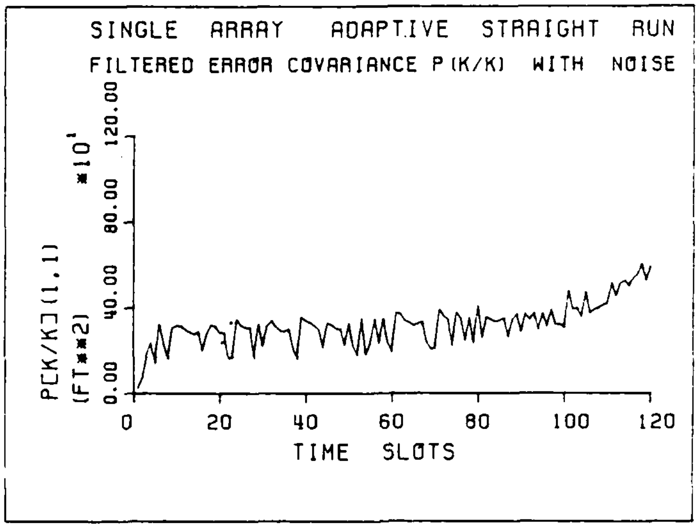


Figure 5.81 Variance of Filtered Position Error in X of the Torpedo During a Straight Run through Single Array

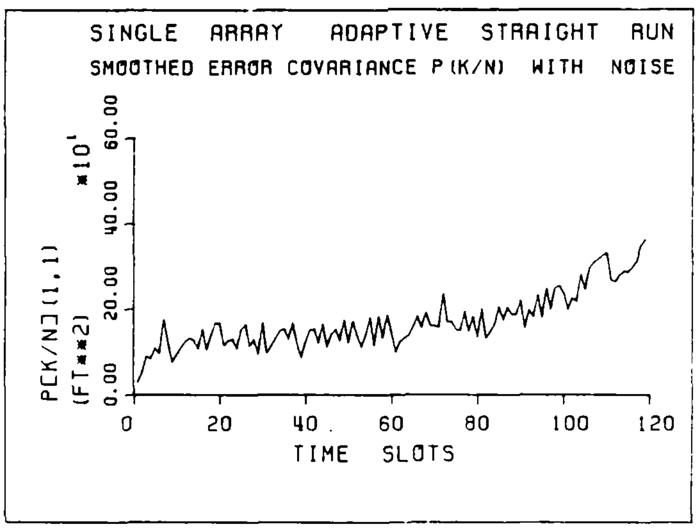


Figure 5.82 Variance of Smoothed Position Error in X of the Torpedo During a Straight Run through Single Array

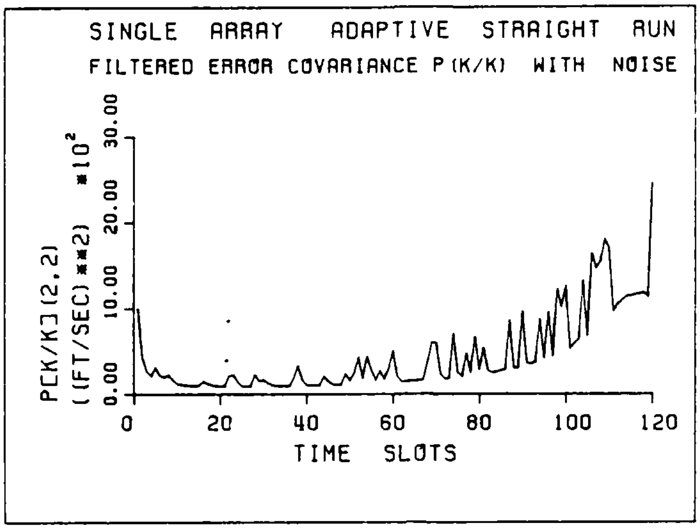


Figure 5.83 Variance of Filtered Velocity Error in X of the Torpedo During a Straight Run through Single Array

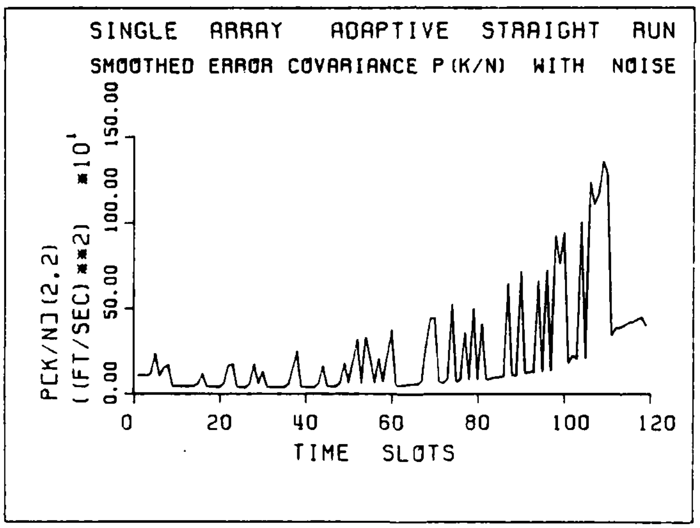


Figure 5.84 Variance of Smoothed Velocity Error in X of the Torpedo During a Straight Run through Single Array

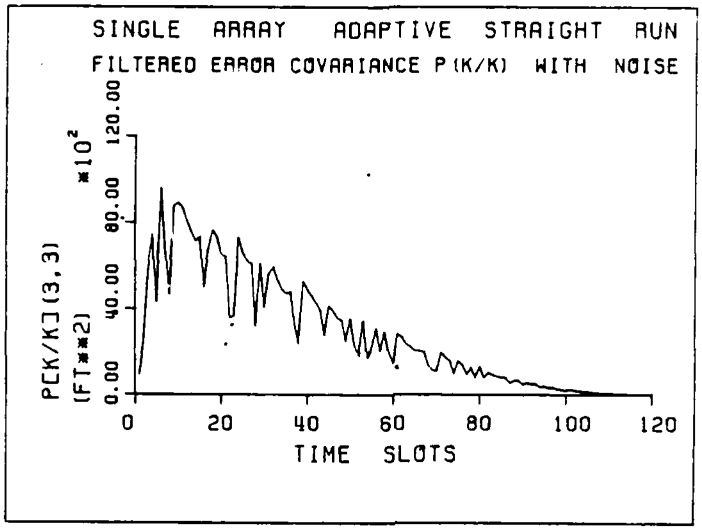


Figure 5.85 Variance of Filtered Position Error in Y of the Torpedo During a Straight Run through Single Array

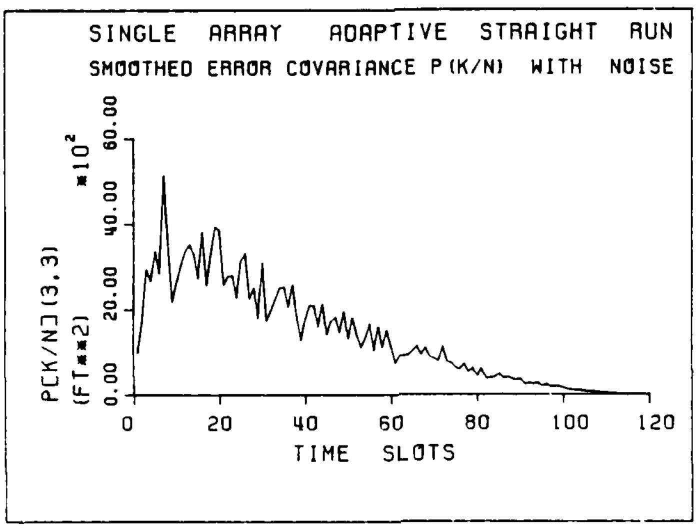


Figure 5.86 Variance of Smoothed Position Error in Y of the Torpedo During a Straight Run through Single Array

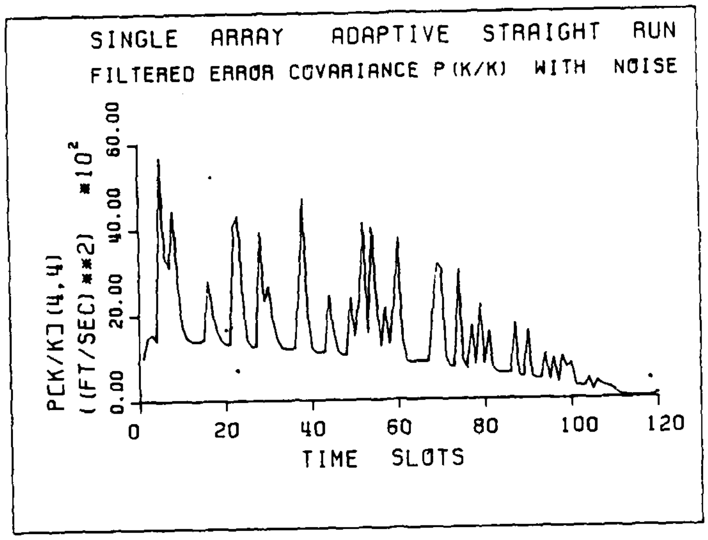


Figure 5.87 Variance of Filtered Velocity Error in Y of the Torpedo During a Straight Run through Single Array

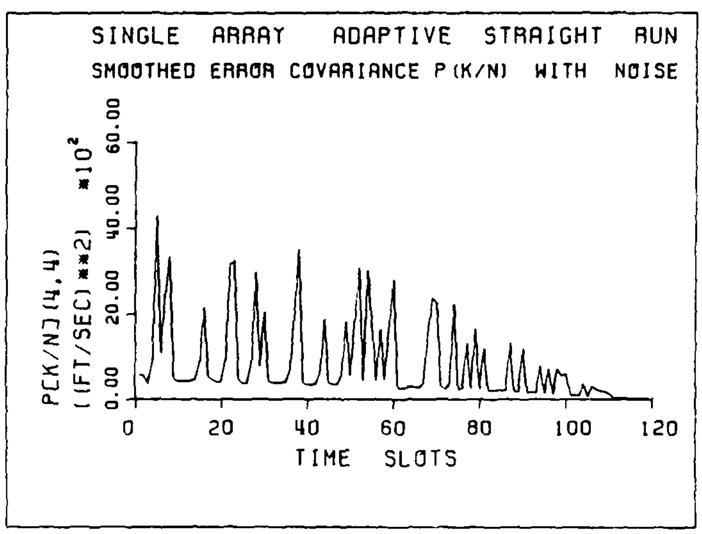


Figure 5.88 Variance of Smoothed Velocity Error in Y of the Torpedo During a Straight Run through Single Array

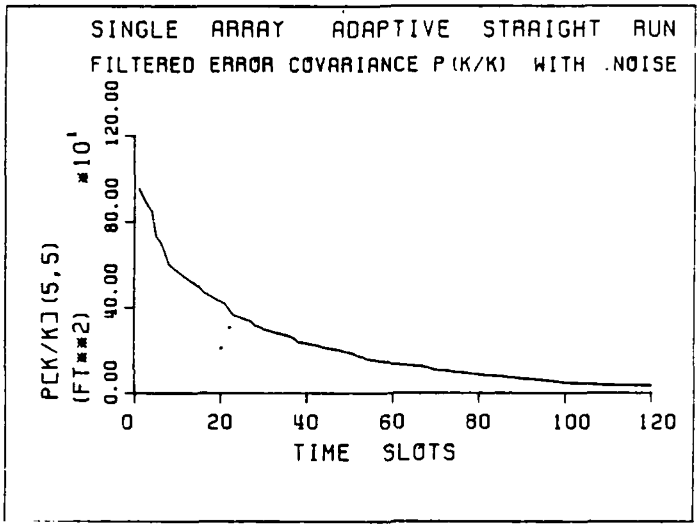


Figure 5.89 Variance of Filtered Position Error in Z of the Torpedo During a Straight Run through Single Array

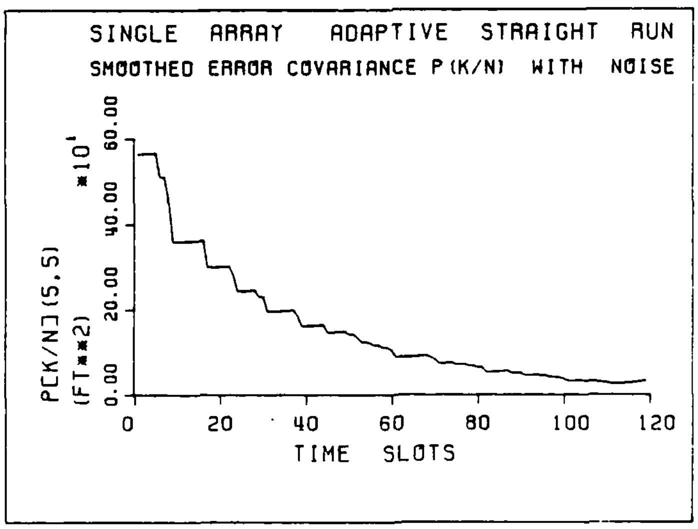


Figure 5.90 Variance of Smoothed Position Error in Z of the Torpedo During a Straight Run through Single Array

APPENDIX A
PROGRAM DESCRIPTION

A. GENERAL

The sequential extended Kalman filter and Smoothing routine is described in detail by [Ref. 1]. Implementation is done by using FORTRAN77 compilers on IBM-PC. [Ref. 10, 11, 12, 13].

B. RUNNING THE PROGRAM ON THE IBM-PC

These directions apply for the IBM-PC computer or other computers(compatibles) with two floopy disk-drives, 640K memory, color/graphic board, math coprocessor and paralel dot matrix printer or printer/plotter. The software utilized during the simulation studies are:

1. Operating System DOS 2.10 with required files to create virtual disk and full screen editor utilities.
2. IBM Professional FORTRAN Compiler 1.00.
3. Microsoft FORTRAN77 3.20.
4. Plotworks PLOT88.LIB.
5. Source files.

Getting the sequential extended Kalman filter and smoothing routine started is essentially a five step process: start your computer; edit the source file and make required changes and then compile; run the executable file

and get the data to be available for plotting routine; edit the source file of plotting routine and make the necessary changes for plotting titles and then compile; run plotting routine. Start the computer up with an operating system and get the program running simply by typing "RUN", which is given in Appendix E, at promt "A>".

APPENDIX B

SEQUENTIAL EXTENDED KALMAN FILTER AND OPTIMAL SMOOTHING PROGRAM LISTING

```
      PROGRAM THESIS
C
C
      REAL*8 XKKM1(5),PKKM1(5,5),PHI(5,5),GAMMA(5,3),GATE
      REAL*8 GAMMAT(3,5),COVW(3,3),COVV(4,4),QTEMP(5,3),P
      REAL*8 TRUX(121),TRUY(121),TRUZ(121),ZI(4),HROW(5)
      REAL*8 ZHAT,GDENOM,GDTEMP,PDUM(5,5),PI(5,5),WHTN,A14
      REAL*8 ZDIFF(4),ZIC(4),XI(5),XKK(5),PKK(5,5),ZDIFAV
      REAL*8 DATR(17),WINIT,PHIPKK(5,5),PKTEMP(5,5),SIGACC
      REAL*8 SIGDIV,SIGCC,XKERR(121),YKERR(121),XP6(5,121)
      REAL*8 HYDRO(6,12),XB(4),YB(4),ZB(4),XSERR(121),TD(3)
      REAL*8 SIGCCC,SIGAAC,SIGDDI,XP(5,121),SMTH(121),GI(5)
      REAL*8 P5(121,5,5),SS1(121,5,5),P1(121,5,5),Q(5,5)
      REAL*8 YSERR(121),ZSERR(121),GNUM(5),PHIT(5,5),XP1(5)
      REAL*8 ZDIFTO,CH(5,5),TEMP1(5,5),XNNM1(5),TEMP2(5)
      REAL*8 AK(5,5),AKT(5,5),TEMP3(5),TEMP4(5,5),XKKS(5)
      REAL*8 PNNM1(5,5),TEMP5(5,5),TEMP6(5,5),PKKS(5,5)
      REAL*8 SS2(5),P2(5,5),SS3(5,5),SS3R(5,5),SIG
      REAL*8 ZKERR(121),X1KERR,X2KERR,Y1KERR,Y2KERR,Z1KERR
      REAL*8 Z2KERR,X1SERR,X2SERR,Y1SERR,Y2SERR,Z1SERR
      REAL*8 Z2SERR
C COORDINATES OF HYDROPHONE ARRAY, FOR MULTIPLE ARRAY
      DATA HYDRO/36000.,30000.,24000.,18000.,12000.,6000.
     +  ,6*6000.,6*0.0,36030.,30030.,24030.,18030.,12030.
     +  ,6030.,6*6000.,6*0.0,36000.,30000.,24000.,18000.
     +  ,12000.,6000.,6*6030.,6*0.0,36000.,30000.,24000.
     +  ,18000.,12000.,7*6000.,6*30./
C
      DATA PKKM1/1000.0,5*0.0,1000.0,5*0.0,1000.0,5*0.0
     +          ,1000.0,5*0.0,1000.0/
      DATA PHI/1.,4*0.,1.31,1.,5*0.,1.,4*0.,1.31,1.,5*0.
     +                                                 ,1./
      DATA GAMMA/0.858,1.31,5*0.0,0.858,1.31,5*0.0,1.31/
      DATA COVW/1.0,3*0.0,1.0,3*0.0,1.0/,WINIT/0.49/
      DATA COVV/1.0D-8,4*0.0,1.0D-8,4*0.0,1.0D-8,4*0.0
     +                                             ,1.0D-8/
C DATA FOR MULTIPLE ARRAY TRACKING
      DATA DATR/38000.,7000.,300.,-50.,0.,3*0.,3*0.
     +,4.712389,.1745329,.1,8.1,600.,800./
      DATA XKKM1/37975.0,-50.0,6975.0,0.0,300.0/
C SECOND DATA FOR MULTIPLE ARRAY TRACKING
C       DATA DATR/35000.,7000.,300.,-50.,0.,3*0.,3*0.
C      +,4.712389,.1745329,.1,8.1,600.,800./
C       DATA XKKM1/34975.0,-50.0,6975.0,0.0,300.0/
```

```
C FIRST DATA FOR SINGLE ARRAY TRACKING
C       DATA DATR/7500.0,1300.0,0.0,-50.0,0.0,3*0.0,3*0.0
C      +,4.712389,.1745329,0.1,8.1,600.0,800.0/
C       DATA XKKM1/7475.0,-50.0,1275.0,0.0,0.0/
C SECOND DATA FOR SINGLE ARRAY TRACKING
C       DATA DATR/2000.0,1000.0,300.0,-50.0,0.0,3*0.0,3*0.0
C      +,4.712389,.1745329,0.1,8.1,600.0,800.0/
C       DATA XKKM1/1975.0,-50.0,975.0,0.0,300.0/
C DATA FOR SUBROUTINE QFIND
      DATA SIGACC/36.2/,SIGDIV/1.0/,SIGCC/22.2/,NZDIFF/4/
      DATA II/1/,IJ/2/,IK/3/,IL/4/,IM/5/,MINE/1/,JTIME/119/
C OPEN STATEMENTS FOR OUTPUT FILES
      OPEN(4,FILE='TRUDI.DAT')
      OPEN(13,FILE='PKK.DAT')
      OPEN(11,FILE='PKN.DAT')
      OPEN(7,FILE='XKK.DAT')
      OPEN(8,FILE='XKN.DAT')
      OPEN(9,FILE='XKERR.DAT')
      OPEN(10,FILE='XSERR.DAT')
      OPEN(12,FILE='OUTPUT.DAT')
C
      SIGCC = (SIGCC * 3.141592654) / 180.0
C TRANSPOSE OF GAMMA MATRIX
      CALL TRANS(GAMMA,IM,IK,GAMMAT)
C TRANSPOSE OF PHI MATRIX
      CALL TRANS(PHI,IM,IM,PHIT)
C USE THESE STATEMENTS TO CALCULATE CONSTANT Q - MATRIX
C       CALL PROD(GAMMA,COVW,IM,IK,IK,QTEMP)
C       CALL PROD(QTEMP,GAMMAT,IM,IK,IM,Q)
      ITIME = JTIME + 1
C USE THIS STATEMENT FOR MULTIPLE ARRAY TRACKING
      I7 = 0
C*****************************************************************
C TIME SLOTS START HERE
      DO 128 KK = 1 , ITIME
       WRITE(*,562) KK
 562   FORMAT(/,10X,'TIME SLOT IN FILTERING :',I5)
C
C&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
C CHOSE THE HYDROPHONE ARRAY FOR MULTIPLE ARRAY TRACKING
       IF(XKKM1(1).GE.33000.0) I8 = 1
       IF((XKKM1(1).GE.27000.0).AND.(XKKM1(1).LT.33000.0))
     +     I8 = 2
       IF((XKKM1(1).GE.21000.0).AND.(XKKM1(1).LT.27000.0))
     +     I8 = 3
       IF((XKKM1(1).GE.15000.0).AND.(XKKM1(1).LT.21000.0))
     +     I8 = 4
       IF((XKKM1(1).GE.9000.0).AND.(XKKM1(1).LT.15000.0))
     +     I8 = 5
       IF(XKKM1(1).LT.9000.0) I8 = 6
C
```

```
 207    DO 205 I3 = 1 , IL
         I4 = 3 * I3
         I5 = I4 - 2
         I6 = I4 - 1
         XB(I3) = HYDRO(I8,I5)
         YB(I3) = HYDRO(I8,I6)
         ZB(I3) = HYDRO(I8,I4)
 205    CONTINUE
C WRITE THE COORDINATES OF CHOSEN HYDROPHONE ARRAY
        IF(I7.NE.I8) THEN
         WRITE(*,217) I8,KK
 217     FORMAT(/,10X,'ARRAY ',I2
     +              ,' STARTS TRACKING AT TIME ',I3)
         WRITE(*,216) KK,I8,(I3,XB(I3),YB(I3)
     +                    ,ZB(I3),I3 = 1 , IL)
 216     FORMAT(I5,I5,4(T11,I5,3(2X,D14.8),/))
         I7 = I8
        ENDIF
C&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
C^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
C CALCULATE THE TRUE TIMES AND THE TRUE TRAJECTORY
C USE THIS CALL STATEMENT FOR MULTIPLE ARRAY TRACKING
        CALL TRJC3(KK,DATR,ZI,TD,XB,YB,ZB)
C USE THIS CALL STATEMENT FOR SINGLE ARRAY TRACKING
C        CALL TRAJEC(KK,DATR,ZI,TD)
        A14 = PHI(1,2)
        TRUX(KK) = TD(1)
        TRUY(KK) = TD(2)
        TRUZ(KK) = TD(3)
C
        WRITE(4,306) KK,TRUX(KK),TRUY(KK),TRUZ(KK)
C^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C                        ROW OF H - MATRIX
        MINE = 1
 163    DO 132 IROW = 1 , IL
         NZDIFF = 4
C USE THIS CALL STATEMENT TO RUN NOISE SUBROUTINE
         CALL NOISE(WINIT,WHTN)
         WHTN = ( 1.0 / 3.0 ) * WHTN
C ZERO NOISE
C         WHTN = 0.0
C USE THIS CALL STATEMENT FOR SINGLE ARRAY TRACKING
C         CALL CHROW(IROW,XKKM1,HROW)
C USE THIS CALL STATEMENT FOR MULTIPLE ARRAY TRACKING
         CALL CHROW3(IROW,XKKM1,HROW,XB,YB,ZB)
C############################################################
C G [ K ] = { P [ K / K - 1 ](5x5) * HT(5x1) } / { H(1x5)
C            * P [ K / K - 1 ](5x5) * HT(5x1) + COV[ V ](1) }
         CALL MMULT(PKKM1,HROW,IM,IM,GNUM)
         CALL VMULT(HROW,GNUM,IM,GDTEMP)
```

```
      GDENOM = GDTEMP + COVV(IROW,IROW)
      DO 134 IX = 1 , IM
       GI(IX) = GNUM(IX) / GDENOM
134   CONTINUE
C###########################################################
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C P [ K / K ] = ( I(5x5) - G [ K ](5x1) * H (1x5) )
C                                          * P [ K / K -1 ]
      DO 135 IP = 1 , IM
       DO 136 JP = 1 , IM
        PDUM(IP,JP) = ( -1. * GI(IP)) * HROW(JP)
        IF(IP.EQ.JP) PDUM(IP,JP) = 1. + PDUM(IP,JP)
136    CONTINUE
135   CONTINUE
      CALL PROD(PDUM,PKKM1,IM,IM,IM,PI)
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C CALCULATE THE PREDICTION OF MEASUREMENTS
C USE THIS CALL STATEMENT FOR SINGLE ARRAY TRACKING
C      CALL CZHAT(IROW,XKKM1,ZHAT)
C USE THIS CALL STATEMENT FOR MULTIPLE ARRAY TRACKING
      CALL CZHAT3(IROW,XKKM1,ZHAT,XB,YB,ZB)
      ZIC(IROW) = ZI(IROW) + WHTN * 0.00001
      ZDIFF(IROW) = ZIC(IROW) - ZHAT
C THREE SIGMA GATE
      P=DMAX1(DABS(PI(1,1)),DABS(PI(3,3)),DABS(PI(5,5)))
      SIG=DSQRT((P/((4860.)**2))+(DABS(COVV(IROW,IROW))))
      GATE = 3.0 * SIG
C
      IF(KK.LE.4) GO TO 149
      IF(DABS(ZDIFF(IROW)).LT.GATE) GO TO 149
C
      WRITE(*,147) KK,IROW,GATE
147   FORMAT(//,10X,'THREE SIGMA GATE HAS BEEN EXCEEDED'
    +      ,' AT TIME ',I4,' IN ROW ',I2,' GATE : ',D14.8)
      DO 148 LGJ = 1 , IM
       GI(LGJ) = 0.0
148   CONTINUE
C TAG INVALID TIME MEASUREMENT
      ZDIFF(IROW) = 999.
C
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
CX [ K / K ] = X [ K / K - 1 ] + G [ K ] * (Z [ K ]
C                                        - Z [ K / K - 1 ])
149   DO 150 I = 1 , IM
       XI(I) = XKKM1(I) + GI(I) * ZDIFF(IROW)
150   CONTINUE
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
      IF(IROW.EQ.4) GO TO 152
      DO 153 I = 1 , IM
       XKKM1(I) = XI(I)
153   CONTINUE
```

```
         DO 155 I = 1 , IM
          DO 154 J = 1 , IM
           PKKM1(I,J) = PI(I,J)
 154      CONTINUE
 155     CONTINUE
 132    CONTINUE
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
 152    DO 156 I = 1, IM
         XKK(I) = XI(I)
         XKKM1(I) = XI(I)
C USE THIS ADDITIONAL STATEMENT FOR SMOOTHING
         XP6(I,KK) = XI(I)
C
         DO 157 J = 1 , IM
          PKK(I,J) = PI(I,J)
          PKKM1(I,J) = PI(I,J)
C USE THIS ADDITIONAL STATEMENT FOR SMOOTHING
          P5(KK,I,J) = PI(I,J)
C
 157     CONTINUE
 156    CONTINUE
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C PREDICTION OF MEASUREMENTS BASED ON X[K/K]
        DO 158 I = 1 , IL
C EDIT INVALID TIME MEASUREMENTS
         IF(ZDIFF(I).GE.999.) GO TO 159
C USE THIS CALL STATEMENT FOR SINGLE ARRAY TRACKING
C         CALL CZHAT(I,XKKM1,ZHAT)
C USE THIS CALL STATEMENT FOR MULTIPLE ARRAY TRACKING
         CALL CZHAT3(I,XKKM1,ZHAT,XB,YB,ZB)
         ZDIFF(I) = DABS(ZIC(I) - ZHAT)
         GO TO 158
 159     ZDIFF(I) = 0.
         NZDIFF = NZDIFF -1
 158    CONTINUE
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
C ABSOLUTE AVERAGE VALUE OF THE DIFFERENCES IN MEASUREMENT
        IF(NZDIFF.EQ.0) GO TO 160
        ZDIFTO = DABS(ZDIFF(1)+ZDIFF(2)+ZDIFF(3)+ZDIFF(4))
        ZDIFAV = ZDIFTO / NZDIFF
        IF(KK.LE.4) GO TO 160
        IF(MINE.EQ.1) SMTH(KK) = ZDIFAV
        IF(MINE.GT.3) GO TO 160
C USE THIS CONSTANT (2.0D-6) GATE
        IF(ZDIFAV.LT.2.0D-6) GO TO 160
C
        WRITE(*,1473) KK
 1473   FORMAT(//,10X,'CONSTANT GATE HAS BEEN EXCEEDED AT'
     +           ,' TIME ',I4)
C((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((
C USE THESE STATEMENT TO INCREASE THE GAIN IN MULTIPLE &
```

```
C SINGLE ARRAY TRACKING
      SIGAAC = 3.0 * SIGACC
      SIGDDI = 3.0 * SIGDIV
      SIGCCC = 3.0 * SIGCC
C
C USE THIS CALL STATEMENT TO CALCULATE ADAPTIVE Q-MATRIX
      CALL QFIND(KK,XKK,PKK,SIGAAC,SIGDDI,SIGCCC,A14,Q)
      CALL ADD(PKK,Q,IM,IM,PKKM1)
      MINE = MINE + 1
      GO TO 163
C((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((
 160  MINE = 1
      NZDIFF = 4
C
      WRITE(7,301) KK,(XKK(J),J = 1 , IM)
      WRITE(13,301) KK,(PKK(I,I),I = 1 , IM)
 301  FORMAT(I5,5(4X,D14.8))
C
      XKERR(KK) = XKK(1) - TRUX(KK)
      YKERR(KK) = XKK(3) - TRUY(KK)
      ZKERR(KK) = XKK(5) - TRUZ(KK)
C
      WRITE(9,306) KK,XKERR(KK),YKERR(KK),ZKERR(KK)
 306  FORMAT(I5,3(4X,D14.8))
C DETERMINE MAX & MIN ERRORS AND THE TIME SLOTS
      IF(KK.EQ.1) THEN
       KX1K = KK
       KX2K = KK
       KY1K = KK
       KY2K = KK
       KZ1K = KK
       KZ2K = KK
       X1KERR = XKERR(KK)
       X2KERR = XKERR(KK)
       Y1KERR = YKERR(KK)
       Y2KERR = YKERR(KK)
       Z1KERR = ZKERR(KK)
       Z2KERR = ZKERR(KK)
      ENDIF
      IF(XKERR(KK).GT.X1KERR) KX1K = KK
      IF(XKERR(KK).GT.X1KERR) X1KERR = XKERR(KK)
      IF(XKERR(KK).LT.X2KERR) KX2K = KK
      IF(XKERR(KK).LT.X2KERR) X2KERR = XKERR(KK)
      IF(YKERR(KK).GT.Y1KERR) KY1K = KK
      IF(YKERR(KK).GT.Y1KERR) Y1KERR = YKERR(KK)
      IF(YKERR(KK).LT.Y2KERR) KY2K = KK
      IF(YKERR(KK).LT.Y2KERR) Y2KERR = YKERR(KK)
      IF(ZKERR(KK).GT.Z1KERR) KZ1K = KK
      IF(ZKERR(KK).GT.Z1KERR) Z1KERR = ZKERR(KK)
      IF(ZKERR(KK).LT.Z2KERR) KZ2K = KK
      IF(ZKERR(KK).LT.Z2KER?) Z2KERR = ZKERR(KK)
```

```
C))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
C P [ K + 1 / K ]=(PHI(5x5) * P [ K / K ](5x5) * PHIT(5x5))
C                                                   + Q [ K ]
C USE THIS CALL STATEMENT TO CALCULATE ADAPTIVE Q-MATRIX
       CALL QFIND(KK,XKK,PKK,SIGACC,SIGDIV,SIGCC,A14,Q)
       CALL PROD(PHI,PKK,IM,IM,IM,PHIPKK)
       CALL PROD(PHIPKK,PHIT,IM,IM,IM,PKTEMP)
       CALL ADD(PKTEMP,Q,IM,IM,PKKM1)
C
       CALL MMULT(PHI,XKK,IM,IM,XKKM1)
C))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
C USE THESE STATEMENTS FOR SMOOTHING
       DO 302 IG = 1 , IM
        XP(IG,KK) = XKK(IG)
 302   CONTINUE
       DO 303 III = 1 , IM
        DO 304 JJJ = 1 , IM
         SS1(KK,III,JJJ) = PKKM1(III,JJJ)
         P1(KK,III,JJJ) = PKK(III,JJJ)
 304    CONTINUE
 303   CONTINUE
C############################################################
C SMOOTHING STARTS HERE
       IF(KK.LE.JTIME) GO TO 128
       DO 500 K = 1 , JTIME
        KI = JTIME - K + 1
        WRITE(*,561) KI
 561    FORMAT(/,10X,'IN SMOOTHING AT TIME :',I5)
        DO 501 I = 1 , IM
         XP1(I) = XP6(I,KI)
 501    CONTINUE
        DO 502 I = 1 , IM
         DO 503 J = 1 , IM
          P2(I,J) = P5(KI,I,J)
          SS3(I,J) = SS1(KI,I,J)
          IF(KI.LE.4) GO TO 503
          IF(SMTH(KI).GE.2.0D-6) SS3(I,J) = 3.6 * SS3(I,J)
 503     CONTINUE
 502    CONTINUE
C************************************************************
C A(K) = P(K/K) * TRANSPOSE[PHI] * INV[P(K+1/K)]
        CALL TRANS(PHI,IM,IM,PHIT)
        CALL RECIP(SS3,IM,SS3R)
        CALL PROD(SS3,SS3R,IM,IM,IM,CH)
        CALL PROD(PHIT,SS3R,IM,IM,IM,TEMP1)
        CALL PROD(P2,TEMP1,IM,IM,IM,AK)
C************************************************************
C X(K/N) = X(K/K) + A(K) * [ X(K+1/N) - X(K+1/K) ]
        DO 504 I = 1 , IM
         XNNM1(I) = XP(I,KI+1)
 504    CONTINUE
```

```
      CALL MMULT(PHI,XP1,IM,IM,SS2)
      CALL SUB(XNNM1,SS2,IM,1,TEMP2)
      CALL PROD(AK,TEMP2,IM,IM,1,TEMP3)
      CALL ADD(XP1,TEMP3,IM,1,XKKS)
      DO 505 I = 1 , IM
       XP(I,KI) = XKKS(I)
 505  CONTINUE
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
      WRITE(8,301) KI,(XKKS(J),J = 1 , IM)
      XSERR(KI) = XKKS(1) - TRUX(KI)
      YSERR(KI) = XKKS(3) - TRUY(KI)
      ZSERR(KI) = XKKS(5) - TRUZ(KI)
      WRITE(10,306) KI,XSERR(KI),YSERR(KI),ZSERR(KI)
C DETERMINE MAX & MIN ERRORS AND THE TIME SLOTS
      IF(K.EQ.1) THEN
       KX1S = KI
       KX2S = KI
       KY1S = KI
       KY2S = KI
       KZ1S = KI
       KZ2S = KI
       X1SERR = XSERR(KI)
       X2SERR = XSERR(KI)
       Y1SERR = YSERR(KI)
       Y2SERR = YSERR(KI)
       Z1SERR = ZSERR(KI)
       Z2SERR = ZSERR(KI)
      ENDIF
      IF(XSERR(KI).GT.X1SERR) KX1S = KI
      IF(XSERR(KI).GT.X1SERR) X1SERR = XSERR(KI)
      IF(XSERR(KI).LT.X2SERR) KX2S = KI
      IF(XSERR(KI).LT.X2SERR) X2SERR = XSERR(KI)
      IF(YSERR(KI).GT.Y1SERR) KY1S = KI
      IF(YSERR(KI).GT.Y1SERR) Y1SERR = YSERR(KI)
      IF(YSERR(KI).LT.Y2SERR) KY2S = KI
      IF(YSERR(KI).LT.Y2SERR) Y2SERR = YSERR(KI)
      IF(ZSERR(KI).GT.Z1SERR) KZ1S = KI
      IF(ZSERR(KI).GT.Z1SERR) Z1SERR = ZSERR(KI)
      IF(ZSERR(KI).LT.Z2SERR) KZ2S = KI
      IF(ZSERR(KI).LT.Z2SERR) Z2SERR = ZSERR(KI)
C**********************************************************************
C P(K/N) = P(K/K)+A(K)*[P(K+1/N)-P(K+1/K)]*TRANSPOSE[A(K)]
      DO 506 I = 1 , IM
       DO 507 J = 1 , IM
        PNNM1(I,J) = P1(KI+1,I,J)
 507   CONTINUE
 506  CONTINUE
      CALL SUB(PNNM1,SS3,IM,IM,TEMP4)
      CALL TRANS(AK,IM,IM,AKT)
      CALL PROD(TEMP4,AKT,IM,IM,IM,TEMP5)
      CALL PROD(AK,TEMP5,IM,IM,IM,TEMP6)
```

```
          CALL ADD(P2,TEMP6,IM,IM,PKKS)
          DO 508 I = 1 , IM
           DO 509 J = 1 , IM
            P1(KI,I,J) = PKKS(I,J)
 509       CONTINUE
 508      CONTINUE
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
          WRITE(11,301) KI,(PKKS(I,I),I = 1 , IM)
C*********************************************************************
 500     CONTINUE
C#####################################################################
 128    CONTINUE
        WRITE(12,800)
 800    FORMAT(10X,' TIME',4X,'   MAX. ERROR   ',4X,' TIME',4X
     +                         ,'   MIN. ERROR   ')
        WRITE(12,801) KX1K,X1KERR,KX2K,X2KERR,KY1K,Y1KERR,
     +KY2K,Y2KERR,KZ1K,Z1KERR,KZ2K,Z2KERR
        WRITE(12,801) KX1S,X1SERR,KX2S,X2SERR,KY1S,Y1SERR,
     +KY2S,Y2SERR,KZ1S,Z1SERR,KZ2S,Z2SERR
 801    FORMAT(3(/,10X,I5,4X,D14.8,4X,I5,4X,D14.8))
        STOP
        END
C
        SUBROUTINE TRANS(AA,NR,NC,BB)
        REAL*8 AA(NR,NC),BB(NC,NR)
        DO 3 I = 1 , NR
         DO 30 J = 1 , NC
          BB(J,I) = AA(I,J)
 30      CONTINUE
 3      CONTINUE
        RETURN
        END
C
        SUBROUTINE PROD(AA,BB,NRA,NCA,NCB,CC)
        REAL*8 AA(NRA,NCA),BB(NCA,NCB),CC(NRA,NCB)
        DO 4 I = 1 , NRA
         DO 40 J = 1 , NCB
          CC(I,J) = 0.0
 40      CONTINUE
 4      CONTINUE
        DO 41 I = 1 , NRA
         DO 410 J = 1 , NCB
          DO 411 K = 1 , NCA
           CC(I,J) = CC(I,J) + AA(I,K) * BB(K,J)
 411      CONTINUE
 410     CONTINUE
 41     CONTINUE
        RETURN
        END
C
        SUBROUTINE TRAJEC(KK,DATR,ZI,TD)
```

```
      REAL*8 DATR(17),ZI(4),TD(3),COEFF,RANGE,VEL,T
      DATA VEL/4860.0/,IIK/3/,IIM/5/
      T = 0.0
      COEFF = 1.0 / VEL
      ZI(1)=COEFF*DSQRT(((DATR(1)+15.0)**2)
     +            +((DATR(2)+15.0)**2)+((DATR(3)+15.0)**2))
      ZI(2)=COEFF*DSQRT(((DATR(1)-15.0)**2)
     +            +((DATR(2)+15.0)**2)+((DATR(3)+15.0)**2))
      ZI(3)=COEFF*DSQRT(((DATR(1)+15.0)**2)
     +            +((DATR(2)-15.0)**2)+((DATR(3)+15.0)**2))
      ZI(4)=COEFF*DSQRT(((DATR(1)+15.0)**2)
     +            +((DATR(2)+15.0)**2)+((DATR(3)-15.0)**2))
      DO 5 I = 1 , IIK
       TD(I) = DATR(I)
 5    CONTINUE
C USE THIS STATEMENT FOR STRAIGHT RUN
C      IF((KK.LE.DATR(17)).AND.(KK.GT.DATR(16))) GO TO 50
C USE THESE STATEMENTS FOR MANEUVERING RUN
 58   IF((KK.LE.49).AND.(KK.GT.22)) GO TO 50
      IF((KK.LE.98).AND.(KK.GT.71)) GO TO 50
      IF((KK.EQ.50).OR.(KK.EQ.99)) THEN
C
C FIRST DATA FOR TRUE TRAJECTORY IN SINGLE ARRAY TRACKING
C       DATR(2) = 1300.0
C       DATR(3) = 0.0
C
C SECOND DATA FOR TRUE TRAJECTORY IN SINGLE ARRAY TACKING
       DATR(2) = 1000.0
       DATR(3) = 300.0
       DATR(4) = -50.0
       DATR(5) = 0.0
       DATR(6) = 0.0
       DATR(7) = 0.0
       DATR(8) = 0.0
       DATR(9) = 0.0
       DATR(10) = 0.0
       DATR(11) = 0.0
       DATR(12) = 4.712389
       DATR(13) = 0.1745329
       DATR(14) =0.1
       DATR(15) = 8.1
      ENDIF
C
 57   DATR(7) = 0.0
      DATR(8) = 0.0
      DATR(14) = 1.31
      GO TO 51
 50   DATR(14) = 0.005
 53   DATR(12) = DATR(12) + DATR(13) * DATR(14)
      DATR(7) = DATR(15) * DCOS(DATR(12))
      DATR(8) = DATR(15) * DSIN(DATR(12))
```

```
 51    DO 52 I = 1 , IIM
        DATR(I) = DATR(I) + DATR(I+3) * DATR(14)
     +                      + (((DATR(14))**2)/2) * DATR(I+6)
 52    CONTINUE
       T = T + DATR(14)
       IF(DABS(T - 1.31).LE.0.0001) RETURN
       GO TO 53
       END
C
       SUBROUTINE TRJC3(KK,DATR,ZI,TD,XB,YB,ZB)
       REAL*8 DATR(17),ZI(4),TD(3),XB(4),YB(4),ZB(4),COEFF
       REAL*8 VEL,T
       DATA VEL/4860.0/,IIK/3/,IIL/4/,IIM/5/
       T = 0.0
       COEFF = 1.0 / VEL
       DO 12 I = 1 , IIL
        ZI(I) = COEFF * DSQRT(((DATR(1) - XB(I))**2)
     +     + ((DATR(2) - YB(I))**2) + ((DATR(3) - ZB(I))**2))
 12    CONTINUE
       DO 120 I = 1 , IIK
        TD(I) = DATR(I)
 120   CONTINUE
C USE THIS STATEMENT FOR  STRAIGHT RUN
       IF((KK.LE.DATR(17)).AND.(KK.GT.DATR(16))) GO TO 121
C USE THESE STATEMENTS FOR MANEUVERING RUN
C 128   IF((KK.LE.49).AND.(KK.GT.22)) GO TO 121
C       IF((KK.LE.98).AND.(KK.GT.71)) GO TO 121
C       IF((KK.EQ.50).OR.(KK.EQ.99)) THEN
C        DATR(2) = 7000.0
C        DATR(3) = 300.0
C        DATR(4) = -50.0
C        DATR(5) = 0.0
C        DATR(6) = 0.0
C        DATR(7) = 0.0
C        DATR(8) = 0.0
C        DATR(9) = 0.0
C        DATR(10) = 0.0
C        DATR(11) = 0.0
C        DATR(12) = 4.712389
C        DATR(13) = 0.1745329
C        DATR(14) = 0.1
C        DATR(15) = 8.1
C       ENDIF
C
 127   DATR(7) = 0.0
       DATR(8) = 0.0
       DATR(14) = 1.31
       GO TO 122
 121   DATR(14) = 0.005
 124   DATR(12) = DATR(12) + DATR(13) * DATR(14)
       DATR(7) = DATR(15) * DCOS(DATR(12))
```

```
      DATR(8) = DATR(15) * DSIN(DATR(12))
 122  DO 123 I = 1 , IIM
       DATR(I) = DATR(I) + DATR(I+3) * DATR(14)
     +                    + (((DATR(14))**2)/2) * DATR(I+6)
 123  CONTINUE
      T = T + DATR(14)
      IF(DABS(T - 1.31).LE.0.0001) RETURN
      GO TO 124
      END
C
      SUBROUTINE CHROW(IROW,XKKM1,HROW)
      REAL*8 HROW(5),XKKM1(5),COEFF,DENOM,DENOM1,DENOM2
      REAL*8 VEL,A1,A2,A3,DENOM3,DENOM4
      DATA VEL/4860.0/
      COEFF = 1.0 / VEL
      DENOM1=DSQRT(((XKKM1(1)+15.0)**2)+((XKKM1(3)+15.0)**2)
     +          +((XKKM1(5)+15.0)**2))
      DENOM2=DSQRT(((XKKM1(1)-15.0)**2)+((XKKM1(3)+15.0)**2)
     +          +((XKKM1(5)+15.0)**2))
      DENOM3=DSQRT(((XKKM1(1)+15.0)**2)+((XKKM1(3)-15.0)**2)
     +          +((XKKM1(5)+15.0)**2))
      DENOM4=DSQRT(((XKKM1(1)+15.0)**2)+((XKKM1(3)+15.0)**2)
     +          +((XKKM1(5)-15.0)**2))
      A1 = 1.0
      A2 = 1.0
      A3 = 1.0
      DENOM = DENOM1
      IF(IROW.EQ.2) DENOM = DENOM2
      IF(IROW.EQ.3) DENOM = DENOM3
      IF(IROW.EQ.4) DENOM = DENOM4
      IF(IROW.EQ.2) A1 = - 1.0
      HROW(1) = COEFF * ((XKKM1(1) + A1 * 15.0) / DENOM)
      IF(IROW.EQ.3) A2 = - 1.0
      HROW(3) = COEFF * ((XKKM1(3) + A2 * 15.0) / DENOM)
      IF(IROW.EQ.4) A3 = - 1.0
      HROW(5) = COEFF * ((XKKM1(5) + A3 * 15.0) / DENOM)
      HROW(2) = 0.0
      HROW(4) = 0.0
      RETURN
      END
C
      SUBROUTINE CHROW3(IROW,XKKM1,HROW,XB,YB,ZB)
      REAL*8 HROW(5),XKKM1(5),COEFF,DENOM,VEL,XB(4),YB(4)
      REAL*8 XO,YO,ZO,ZB(4)
      DATA VEL/4860.0/
      COEFF = 1.0 / VEL
      XO = XB(IROW)
      YO = YB(IROW)
      ZO = ZB(IROW)
      DENOM = DSQRT(((XKKM1(1)-XO)**2)+((XKKM1(3)-YO)**2)
     +                              + ((XKKM1(5)-ZO)**2))
```

```
      HROW(1) = COEFF * ((XKKM1(1) - XO) / DENOM)
      HROW(2) = 0.0
      HROW(3) = COEFF * ((XKKM1(3) - YO) / DENOM)
      HROW(4) = 0.0
      HROW(5) = COEFF * ((XKKM1(5) - ZO) / DENOM)
      RETURN
      END
C
      SUBROUTINE MMULT(AA,BB,NRA,NCA,CC)
      REAL*8 AA(NRA,NCA),BB(NCA),CC(NRA)
      DO 6 I = 1 , NRA
       CC(I) = 0.0
       DO 60 J = 1 , NCA
        CC(I) = CC(I) + AA(I,J) * BB(J)
 60    CONTINUE
 6    CONTINUE
      RETURN
      END
C
      SUBROUTINE VMULT(AA,BB,NE,CC)
      REAL*8 AA(NE),BB(NE),CC
      CC = 0.0
      DO 7 I = 1 , NE
       CC = CC + AA(I) * BB(I)
 7    CONTINUE
      RETURN
      END
C
      SUBROUTINE CZHAT(IROW,XKKM1,ZHAT)
      REAL*8 XKKM1(5),ZHAT,COEFF,VEL
      DATA VEL/4860.0/
      COEFF = 1.0 / VEL
      IF(IROW.EQ.1) ZHAT=COEFF*DSQRT(((XKKM1(1)+15.0)**2)+
     +       ((XKKM1(3)+15.0)**2)+((XKKM1(5)+15.0)**2))
      IF(IROW.EQ.2) ZHAT=COEFF*DSQRT(((XKKM1(1)-15.0)**2)+
     +       ((XKKM1(3)+15.0)**2)+((XKKM1(5)+15.0)**2))
      IF(IROW.EQ.3) ZHAT=COEFF*DSQRT(((XKKM1(1)+15.0)**2)+
     +       ((XKKM1(3)-15.0)**2)+((XKKM1(5)+15.0)**2))
      IF(IROW.EQ.4) ZHAT=COEFF*DSQRT(((XKKM1(1)+15.0)**2)+
     +       ((XKKM1(3)+15.0)**2)+((XKKM1(5)-15.0)**2))
      RETURN
      END
C
      SUBROUTINE CZHAT3(IROW,XKKM1,ZHAT,XB,YB,ZB)
      REAL*8 XKKM1(5),ZHAT,COEFF,VEL,XB(4),YB(4),ZB(4),XO
      REAL*8 YO,ZO
      DATA VEL/4860.0/
      COEFF = 1.0 / VEL
      XO = XB(IROW)
      YO = YB(IROW)
      ZO = ZB(IROW)
```

```
      ZHAT = COEFF * DSQRT(((XKKM1(1) - XO)**2) +
     +           ((XKKM1(3) - YO)**2) + ((XKKM1(5) - ZO)**2))
      RETURN
      END
C
      SUBROUTINE NOISE(R,P)
      REAL*8 Y(6),X(6),S(5),R,P,BB,P1
      DATA Y/0.0,.0228,.0668,.1357,.2743,.5/
      DATA X/-3.01,-2.0,-1.5,-1.0,-0.6,0.0/
      DATA S/43.8596,11.3636,7.25689,2.891352,2.65887/
      BB = 1.0
      P1 = R * 317.0
      R = DMOD(P1 , BB)
      P = R
      I = 1
      IF(P.GT.0.5) P = 1.0 - R
 8    IF(P.LT.Y(I+1)) GO TO 80
      I = I + 1
      GO TO 8
 80   P = ((P - Y(I)) * S(I) + X(I))
      IF(R.GE.0.5) P = - P
      RETURN
      END
C
      SUBROUTINE ADD(AA,BB,NR,NC,CC)
      REAL*8 AA(NR,NC),BB(NR,NC),CC(NR,NC)
      DO 9 I = 1 , NR
       DO 90 J = 1 , NC
        CC(I,J) = AA(I,J) + BB(I,J)
 90    CONTINUE
 9    CONTINUE
      RETURN
      END
C
      SUBROUTINE QFIND(KK,XKK,PKK,SIGACC,SIGDIV,SIGCC,A,Q)
      REAL*8 XKK(5),PKK(5,5),Q(5,5),SIGACC,SIGDIV,SIGCC,A
      REAL*8 A2,A3,B,C,D,E1,E12,E2,G1,G2,G3,SIGAAC,SIGDDI
      REAL*8 SIGCCC,A1
      INTEGER KK
      IF(KK.NE.1) GO TO 111
      DO 11 I = 1 , 5
       DO 110 J = 1 , 5
        Q(I,J) = 0.0
 110   CONTINUE
 11   CONTINUE
      SIGACC = SIGACC **2
      Q(5,5) = (SIGDIV **2) * (A **2)
      SIGCC = SIGCC **2
      G1 = ( A **2 ) / 2.0
      G2 = G1 **2
      G3 = A * G1
```

```
      A2 = A **2
 111  A1 = XKK(2) **2 + XKK(4) **2
      A3 = XKK(2) / DSQRT(A1)
      B = XKK(4)
      C = XKK(4) / DSQRT(A1)
      D = XKK(2)
      E1 = ( A3 **2) * SIGACC + ( B **2 ) * SIGCC
      E12 = A3 * C * SIGACC - B * D * SIGCC
      E2 = ( C **2 ) * SIGACC + ( D **2 ) * SIGCC
      Q(1,1) = E1 * G2
      Q(1,2) = G3 * E1
      Q(1,3) = E12 * G2
      Q(1,4) = G3 * E12
      Q(2,2) = A2 * E1
      Q(2,3) = G3 * E12
      Q(2,4) = A2 * E12
      Q(3,3) = G2 * E2
      Q(3,4) = G3 * E2
      Q(4,4) = A2 * E2
      DO 112 I = 1 , 4
       DO 113 J = 1 , I
        Q(I,J) = Q(J,I)
 113   CONTINUE
 112  CONTINUE
      RETURN
      END
C
      SUBROUTINE SUB(AA,BB,NR,NC,CC)
      REAL*8 AA(NR,NC),BB(NR,NC),CC(NR,NC)
      DO 12 I = 1 , NR
       DO 120 J = 1 , NC
        CC(I,J) = AA(I,J) - BB(I,J)
 120   CONTINUE
 12   CONTINUE
      RETURN
      END
C
      SUBROUTINE RECIP(AA,NN,CC)
      REAL*8 AA(NN,NN),DD(5,10),CC(NN,NN)
      DO 14 K = 1 , NN
       DO 140 J = 1 , NN
        DD(K,J) = AA(K,J)
 140   CONTINUE
 14   CONTINUE
      DO 141 K = 1 , NN
       I = K + NN
       DO 142  J = 6 , 10
        IF(I.NE.J) GO TO 143
        DD(K,J) = 1.
        GO TO 142
 143    DD(K,J) = 0.
```

```
142    CONTINUE
141   CONTINUE
      DO 144 K = 1 , NN
       M = K + 1
       DO 145 J = M , 10
        DD(K,J) = DD(K,J) / DD(K,K)
145    CONTINUE
       DD(K,K) = 1.
       DO 146 L = 1 , NN
        IF(L.EQ.K) GO TO 146
        DO 147 I = 1 , 10
         IF(I.EQ.K) GO TO 147
         DD(L,I) = DD(L,I) - DD(L,K) * DD(K,I)
147     CONTINUE
        DD(L,K) = 0.
146    CONTINUE
144   CONTINUE
      DO 148 K = 1 , NN
       DO 149 J = 1 , NN
        I = J + NN
        CC(K,J) = DD(K,I)
149    CONTINUE
148   CONTINUE
      RETURN
      END
```

APPENDIX C

PLOTTING PROGRAM LISTING FOR HP PLOTTER

```
$STORAGE:2
$DEBUG
$NOLIST
C
      PROGRAM PLOTTER
C
      CHARACTER*40 TITLE
      CHARACTER*35 LEGEND,SUBTITLE
      CHARACTER*25 NAMEX,NAMEY
      REAL X(245),Y(245)
      REAL O(245),P(245),R(245),S(245),T(245),U(245)
      INTEGER*2 IC
      DATA IC/0/
C
C USE THESE FOR MULTIPLE ARRAY TRACKING
C       TITLE ='MULTIPLE ARRAY ADAPTIVE MANEUVERING RUN'
      TITLE ='MULTIPLE  ARRAY  ADAPTIVE  STRAIGHT RUN'
C
C USE THESE FOR SINGLE ARRAY TRACKING
C       TITLE ='SINGLE ARRAY  ADAPTIVE  MANEUVERING RUN'
C       TITLE ='SINGLE  ARRAY  ADAPTIVE  STRAIGHT  RUN'
      OPEN(5,FILE='XKK.DAT',STATUS='OLD')
      DO 32 LENG = 1 , 241
       READ(5,*,END=33) O(LENG),P(LENG),R(LENG),S(LENG),
     +                  T(LENG),U(LENG)
32    CONTINUE
33    CONTINUE
      LENG = LENG - 1
      CLOSE(5,STATUS='KEEP')
      NAMEX ='X[K/K]  (FT)'
      NAMEY ='Y[K/K]  (FT)'
      SUBTITLE ='                    '
      LEGEND ='FILTERED ESTIMATE OF TRAJECTORY'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,P,S,LENG,
     +                                  SUBTITLE)
      OPEN(5,FILE='PKK.DAT',STATUS='OLD')
      DO 34 LENG = 1 , 241
      READ(5,*,END=35) O(LENG),P(LENG),R(LENG),S(LENG),
     +                 T(LENG),U(LENG)
34    CONTINUE
35    CONTINUE
      LENG = LENG - 1
      CLOSE(5,STATUS='KEEP')
      NAMEX ='TIME  SLOTS'
      NAMEY ='(FT**2)'
```

```
      SUBTITLE ='P[K/K](1,1)'
      LEGEND ='FILTERED ERROR COVARIANCE P(K/K)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,P,LENG,
     +                                    SUBTITLE)
      NAMEY ='((FT/SEC)**2)'
      SUBTITLE ='P[K/K](2,2)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,R,LENG,
     +                                    SUBTITLE)
      NAMEY ='(FT**2)'
      SUBTITLE ='P[K/K](3,3)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,S,LENG,
     +                                    SUBTITLE)
      NAMEY ='((FT/SEC)**2)'
      SUBTITLE ='P[K/K](4,4)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,T,LENG,
     +                                    SUBTITLE)
      NAMEY ='(FT**2)'
      SUBTITLE ='P[K/K](5,5)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,U,LENG,
     +                                    SUBTITLE)
      OPEN(5,FILE='XKERR.DAT',STATUS='OLD')
      DO 38 LENG = 1 , 241
      READ(5,*,END=39) O(LENG),P(LENG),R(LENG),S(LENG)
 38   CONTINUE
 39   CONTINUE
      LENG = LENG - 1
      CLOSE(5,STATUS='KEEP')
      NAMEY ='X ERROR  (FT)'
      SUBTITLE ='
      LEGEND ='ERROR IN FILTERED ESTIMATE'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,P,LENG,
     +                                    SUBTITLE)
      NAMEY ='Y ERROR  (FT)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,R,LENG,
     +                                     SUBTITLE)
      NAMEY ='Z ERROR  (FT)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,S,LENG,
     +                                     SUBTITLE)
      OPEN(5,FILE='XKN.DAT',STATUS='OLD')
      DO 40 LENG = 1 , 241
      READ(5,*,END=41) O(LENG),P(LENG),R(LENG),S(LENG),
     +                 T(LENG),U(LENG)
 40   CONTINUE
 41   CONTINUE
      LENG = LENG - 1
      CLOSE(5,STATUS='KEEP')
      NAMEX ='X[K/N]  (FT)'
      NAMEY ='Y[K/N]  (FT)'
      LEGEND ='SMOOTHED ESTIMATE OF TRAJECTORY'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,P,S,LENG,
     +                                     SUBTITLE)
```

```
      OPEN(5,FILE='PKN.DAT',STATUS='OLD')
      DO 42 LENG = 1 , 241
      READ(5,*,END=43) O(LENG),P(LENG),R(LENG),S(LENG),
     +                 T(LENG),U(LENG)
42    CONTINUE
43    CONTINUE
      LENG = LENG - 1
      CLOSE(5,STATUS='KEEP')
      NAMEX ='TIME  SLOTS'
      NAMEY ='(FT**2)'
      SUBTITLE ='P[K/N](1,1)'
      LEGEND ='SMOOTHED ERROR COVARIANCE P(K/N)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,P,LENG,
     +                                     SUBTITLE)
      NAMEY ='((FT/SEC)**2)'
      SUBTITLE ='P[K/N](2,2)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,R,LENG,
     +                                     SUBTITLE)
      NAMEY ='(FT**2)'
      SUBTITLE ='P[K/N](3,3)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,S,LENG,
     +                                     SUBTITLE)
      NAMEY ='((FT/SEC)**2)'
      SUBTITLE ='P[K/N](4,4)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,T,LENG,
     +                                     SUBTITLE)
      NAMEY ='(FT**2)'
      SUBTITLE ='P[K/N](5,5)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,U,LENG,
     +                                     SUBTITLE)
      OPEN(5,FILE='XSERR.DAT',STATUS='OLD')
      DO 44 LENG = 1 , 241
      READ(5,*,END=45) O(LENG),P(LENG),R(LENG),S(LENG)
44    CONTINUE
45    CONTINUE
      LENG = LENG - 1
      CLOSE(5,STATUS='KEEP')
      NAMEY ='X ERROR  (FT)'
      SUBTITLE ='                    '
      LEGEND ='ERROR IN SMOOTHED ESTIMATE'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,P,LENG,
     +                                     SUBTITLE)
      NAMEY ='Y ERROR  (FT)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,R,LENG,
     +                                     SUBTITLE)
      NAMEY ='Z ERROR  (FT)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,S,LENG,
     +                                     SUBTITLE)
      STOP
      END
      SUBROUTINE DRAWER(TITLE,NAMEX,NAMEY,LEGEND,X,Y,
```

```
     +                                               LENG,SUBTITLE)
      CHARACTER*40 TITLE
      CHARACTER*35 LEGEND,SUBTITLE
      CHARACTER*25 NAMEX,NAMEY
      REAL X(245),Y(245)
      INTEGER*2 IC
      DATA IC/0/
C
      CALL ED
      CALL CUP(1,0)
      CALL  PLOTS(0,9600,30)
      CALL SYMBOL(2.0,6.65,.20,TITLE,0.0,40)
      CALL SYMBOL(2.0,6.25,.175,LEGEND,0.0,35)
C
C USE THIS FOR NOISELESS TRACKING
C       CALL SYMBOL(6.84,6.25,.175,'WITHOUT NOISE',0.0,13)
C
C USE THIS FOR NOISY TRACKING
      CALL SYMBOL(6.84,6.25,.175,'  WITH  NOISE',0.0,13)
C
      CALL SYMBOL(1.60,2.45,.20,SUBTITLE,90.0,35)
      CALL PLOT(1.00,1.00,-3)
      CALL PLOT(8.0,0.0,3)
      CALL PLOT(8.0,6.0,2)
      CALL PLOT(0.0,6.0,2)
      CALL PLOT(0.0,0.0,2)
      CALL PLOT(8.0,0.0,2)
      CALL SCALE(X,6.00,LENG,1)
      CALL SCALE(Y,3.00,LENG,1)
      CALL STAXIS(.180,.20,.15,.112,-1)
      CALL AXIS(1.5,1.5,NAMEX,-13,6.00,00.,X(LENG+1),
     +                     X(LENG+2))
      CALL STAXIS(.15,.20,.111,.112,2)
      CALL AXIS(1.5,1.5,NAMEY,13,3.00,90.,Y(LENG+1),
     +                     Y(LENG+2))
      CALL PLOT(1.50,1.50,-3)
      CALL LINE(X,Y,LENG,1,0,3)
      CALL PLOT(0.0,0.0,999)
      RETURN
      END
      SUBROUTINE ED
      CHARACTER*1 C1,C2,C3,C4
      INTEGER*2 IC(4)
      EQUIVALENCE (C1,IC(1)),(C2,IC(2)),(C3,IC(3)),
     +                                 (C4,IC(4))
      DATA IC/16#1B,16#5B,16#32,16#4A/
      WRITE(*,1) C1,C2,C3,C4
 1    FORMAT(1X,4A1)
      RETURN
      END
      SUBROUTINE CUP(N,M)
```

```
      CHARACTER*1 C1,C2,C5,C8,LC(5)
      CHARACTER*5 CBUFF
      INTEGER*2 IC(4)
      EQUIVALENCE (C1,IC(1)),(C2,IC(2)),(C5,IC(3)),
     +            (C8,IC(4)),(CBUFF,LC(1))
      DATA IC/16#1B,16#5B,16#3B,16#66/
      L=10000+100*N+M
      WRITE(CBUFF,2)L
 2    FORMAT(I5)
      WRITE(*,1) C1,C2,LC(2),LC(3),C5,LC(4),LC(5),C8
1     FORMAT(1X,8A1,\)
      RETURN
      END
```

APPENDIX D

PLOTTING PROGRAM LISTING FOR MONITOR

```
$STORAGE:2
$DEBUG
$NOLIST
C
      PROGRAM MONITOR
C
      CHARACTER*40 TITLE
      CHARACTER*35 LEGEND
      CHARACTER*25 NAMEX,NAMEY
      REAL X(245),Y(245)
      REAL O(245),P(245),R(245),S(245),T(245),U(245)
      INTEGER*2 IC
      DATA IC/0/
C
C USE THESE FOR MULTIPLE ARRAY TRACKING
C      TITLE ='MULTIPLE ARRAY ADAPTIVE MANEUVERING RUN'
      TITLE ='MULTIPLE  ARRAY  ADAPTIVE  STRAIGHT RUN'
C
C USE THESE FOR SINGLE ARRAY TRACKING
C      TITLE ='SINGLE ARRAY  ADAPTIVE  MANEUVERING RUN'
C      TITLE ='SINGLE  ARRAY   ADAPTIVE  STRAIGHT  RUN'
      OPEN(5,FILE='XKK.DAT',STATUS='OLD')
      DO 32 LENG = 1 , 241
       READ(5,*,END=33) O(LENG),P(LENG),R(LENG),S(LENG),
     +                  T(LENG),U(LENG)
32    CONTINUE
33    CONTINUE
      LENG = LENG - 1
      CLOSE(5,STATUS='KEEP')
      NAMEX ='X[K/K]  (FT)'
      NAMEY ='Y[K/K]  (FT)'
      LEGEND ='FILTERED ESTIMATE OF TRAJECTORY'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,P,S,LENG)
      OPEN(5,FILE='PKK.DAT',STATUS='OLD')
      DO 34 LENG = 1 , 241
      READ(5,*,END=35) O(LENG),P(LENG),R(LENG),S(LENG),
     +                 T(LENG),U(LENG)
34    CONTINUE
35    CONTINUE
      LENG = LENG - 1
      CLOSE(5,STATUS='KEEP')
      NAMEX ='TIME  SLOTS'
      NAMEY ='P[K/K](1,1)'
      LEGEND ='FILTERED ERROR COVARIANCE P(K/K)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,P,LENG)
```

```
      NAMEY ='P[K/K](2,2)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,R,LENG)
      NAMEY ='P[K/K](3,3)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,S,LENG)
      NAMEY ='P[K/K](4,4)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,T,LENG)
      NAMEY ='P[K/K](5,5)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,U,LENG)
      OPEN(5,FILE='XKERR.DAT',STATUS='OLD')
      DO 38 LENG = 1 , 241
      READ(5,*,END=39) O(LENG),P(LENG),R(LENG),S(LENG)
38    CONTINUE
39    CONTINUE
      LENG = LENG - 1
      CLOSE(5,STATUS='KEEP')
      NAMEY ='X ERROR  (FT)'
      LEGEND ='ERROR IN FILTERED ESTIMATE'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,P,LENG)
      NAMEY ='Y ERROR  (FT)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,R,LENG)
      NAMEY ='Z ERROR  (FT)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,S,LENG,
      OPEN(5,FILE='XKN.DAT',STATUS='OLD')
      DO 40 LENG = 1 , 241
      READ(5,*,END=41) O(LENG),P(LENG),R(LENG),S(LENG),
     +                 T(LENG),U(LENG)
40    CONTINUE
41    CONTINUE
      LENG = LENG - 1
      CLOSE(5,STATUS='KEEP')
      NAMEX ='X[K/N]  (FT)'
      NAMEY ='Y[K/N]  (FT)'
      LEGEND ='SMOOTHED ESTIMATE OF TRAJECTORY'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,P,S,LENG)
      OPEN(5,FILE='PKN.DAT',STATUS='OLD')
      DO 42 LENG = 1 , 241
      READ(5,*,END=43) O(LENG),P(LENG),R(LENG),S(LENG),
     +                 T(LENG),U(LENG)
42    CONTINUE
43    CONTINUE
      LENG = LENG - 1
      CLOSE(5,STATUS='KEEP')
      NAMEX ='TIME  SLOTS'
      NAMEY ='P[K/N](1,1)'
      LEGEND ='SMOOTHED ERROR COVARIANCE P(K/N)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,P,LENG)
      NAMEY ='P[K/N](2,2)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,R,LENG)
      NAMEY ='P[K/N](3,3)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,S,LENG)
      NAMEY ='P[K/N](4,4)'
```

```
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,T,LENG)
      NAMEY ='P[K/N](5,5)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,U,LENG)
      OPEN(5,FILE='XSERR.DAT',STATUS='OLD')
      DO 44 LENG = 1 , 241
      READ(5,*,END=45) O(LENG),P(LENG),R(LENG),S(LENG)
44    CONTINUE
45    CONTINUE
      LENG = LENG - 1
      CLOSE(5,STATUS='KEEP')
      NAMEY ='X ERROR  (FT)'
      LEGEND ='ERROR IN SMOOTHED ESTIMATE'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,P,LENG)
      NAMEY ='Y ERROR  (FT)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,R,LENG)
      NAMEY ='Z ERROR  (FT)'
      CALL DRAWER(TITLE,NAMEX,NAMEY,LEGEND,O,S,LENG)
      STOP
      END
      SUBROUTINE DRAWER(TITLE,NAMEX,NAMEY,LEGEND,X,Y,LENG)
      CHARACTER*40 TITLE
      CHARACTER*35 LEGEND
      CHARACTER*25 NAMEX,NAMEY
      REAL X(245),Y(245)
      INTEGER*2 IC
      DATA IC/0/
C
      CALL ED
      CALL CUP(1,0)
      CALL  PLOTS(0,99,99)
      CALL SYMBOL(0.5,5.15,.20,TITLE,0.0,40)
      CALL SYMBOL(1.04,4.75,.175,LEGEND,0.0,35)
C
C USE THIS FOR NOISELESS TRACKING
C       CALL SYMBOL(5.38,4.75,.175,'WITHOUT NOISE',0.0,13)
C
C USE THIS FOR NOISY TRACKING
      CALL SYMBOL(5.38,4.75,.175,'  WITH  NOISE',0.0,13)
C
      CALL PLOT(1.00,1.00,-3)
      CALL SCALE(X,6.00,LENG,1)
      CALL SCALE(Y,3.00,LENG,1)
      CALL STAXIS(.180,.20,.15,.112,-1)
      CALL AXIS(0.,0.,NAMEX,-13,6.00,00.,X(LENG+1)
     +                  X(LENG+2))
      CALL STAXIS(.15,.20,.111,.112,2)
      CALL AXIS(0.,0.,NAMEY,13,3.00,90.,Y(LENG+1),
     +                  Y(LENG+2))
      CALL LINE(X,Y,LENG,1,0,3)
      CALL PLOT(0.0,0.0,999)
      RETURN
```

```
      END
      SUBROUTINE ED
      CHARACTER*1 C1,C2,C3,C4
      INTEGER*2 IC(4)
      EQUIVALENCE (C1,IC(1)),(C2,IC(2)),(C3,IC(3)),
     +                            (C4,IC(4))
      DATA IC/16#1B,16#5B,16#32,16#4A/
      WRITE(*,1) C1,C2,C3,C4
1     FORMAT(1X,4A1)
      RETURN
      END
      SUBROUTINE CUP(N,M)
      CHARACTER*1 C1,C2,C5,C8,LC(5)
      CHARACTER*5 CBUFF
      INTEGER*2 IC(4)
      EQUIVALENCE (C1,IC(1)),(C2,IC(2)),(C5,IC(3)),
     +           (C8,IC(4)),(CBUFF,LC(1))
      DATA IC/16#1B,16#5B,16#3B,16#66/
      L=10000+100*N+M
      WRITE(CBUFF,2)L
 2    FORMAT(I5)
      WRITE(*,1) C1,C2,LC(2),LC(3),C5,LC(4),LC(5),C8
1     FORMAT(1X,8A1,\)
      RETURN
      END
```

APPENDIX E
BATCH FILES

A. LISTING OF AUTOEXEC.BAT FILE ON OPERATING SYSTEM DISK

```
ECHO OFF
GRAPHICS
TIMER/S
COPY A:RUN.BAT C:
COPY A:KEDIT.EXE C:/V
COPY A:PROFILE.KED C:/V
C:
RUN
```

B. LISTING OF RUN.BAT FILE ON VIRTUAL DISK(C)

```
ECHO. Insert the disk, which has the source file of the
ECHO. sequential extended Kalman filter and Smoothing,
ECHO. into drive A
PAUSE
COPY A:THESIS.FOR C:
KEDIT C:THESIS.FOR
COPY C:THESIS.FOR A:
ERASE C:KEDIT.EXE
ERASE C:PROFILE.KED
ECHO. Insert the disk, which has PROFORT.EXE and
ECHO. LINK.EXE, into drive A, and the disk, which has
ECHO. PROFORT.LIB into drive B.
PAUSE
A:PROFORT THESIS /L /E
A:LINK THESIS,,NULL,PROFORT
ERASE C:THESIS.FOR
ERASE C:THESIS.OBJ
THESIS
ECHO. Insert the disk, which has the source file of the
ECHO. sequential extended Kalman filter and Smoothing,
ECHO. into drive A, and the disk labeled "DATA" into
ECHO. drive B.
PAUSE
COPY C:THESIS.EXE A:
COPY C:*.DAT B:
ERASE C:*.*
ECHO. Insert the operating system disk into drive A, and
ECHO. the disk, which has the plotting routine source
ECHO. file into drive B.
PAUSE
COPY A:KEDIT.EXE C:
COPY A:PROFILE.KED C:
COPY B:GRAPH.FOR C:
KEDIT C:GRAPH.FOR
COPY C:GRAPH.FOR B:
```

```
ERASE C:KEDIT.EXE
ERASE C:PROFILE.KED
ECHO. Insert the disk, which has FOR1.EXE and PAS2.EXE
ECHO. into drive A, and the disk, which has PLOT88.LIB
ECHO. into drive B.
PAUSE
A:FOR1 GRAPH;
A:PAS2
ECHO. Insert the disk, which has FORTRAN.LIB, MATH.LIB
ECHO. and LINK.EXE into drive A.
PAUSE
A:LINK GRAPH,,NULL,B:PLOTT88+A:FORTRAN+A:MATH
ECHO. Insert the disk, which has the plotting source
ECHO. file into drive A and the data disk into drive B.
PAUSE
COPY C:GRAPH.EXE A:
ERASE C:GRAPH.FOR
ERASE C:GRAPH.OBJ
COPY B:*.DAT C:
GRAPH
```

LIST OF REFERENCES

1. Isik, M., An Application of Kalman Filtering and Smoothing to Torpedo Tracking, M.S. Thesis, Naval Postgraduate School, Monterey, California, 1983.

2. Technical Manual, NAVORD OD 41964, NAVTOPRSTA Keyport Range Complex and Associated Data, May 1970.

3. Maybeck, P. S., Stochastic Models, Estimation, and Control, Academic Press, 1982.

4. Brown, R. G., Introduction to Random Signal Analysis and Kalman Filtering, Jhon Wiley & Sons,Inc., 1983.

5. Gelb, A., Applied Optimal Estimation, M.I.T. Press, 1974.

6. Rauch, H. E., Tung, F. and Striebel, C. T., Maximum Likelihood Estimates of Linear Dynamic Systems, AIAA JOURNAL, pp. 1445-1450, August 1965.

7. Rauch, H. E., Solution to the Linear Smoothing Problem, IEEE TRANSACTION ON AUTOMATIC CONTROL, pp. 371-372, October 1963.

8. O'Brien, P. A., An Application of Kalman Filtering to Underwater Tracking, M.S. Thesis, Naval Postgraduate School, Monterey, California, September 1980.

9. IBM, Technical Reference for the IBM Personal Computer, Personal Computer Hardware Reference Library, 1983.

10. IBM, Personal Computer Professional FORTRAN Reference, Installation and use, Personal Computer Software, 1984.

11. Microsoft, Microsoft FORTRAN77 Reference Manual, Microsoft Corporation, 1984.

12. Peerles Engineering Service, Fortran Scientific Subroutine Library, John Wiley & Sons, Inc, 1984.

13. Young, T., L., and Van Woert, M., L., PLOT88 Software Library Reference Manual, PLOTWORKS, Inc, 1984.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5000	2
3.	Department Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	2
4.	Professor H. A. Titus, Code 62Ts Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	5
5.	Associate Professor A. Gerba, Code 62Gz Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	1
6.	Commanding Officer Naval Underseas Weapons Engineering Station Keyport, Washington 98345	2
7.	Deniz Kuvvetleri Komutanligi Kutuphanesi Bakanliklar - ANKARA / TURKEY	5
8.	Deniz Harp Okulu Komutanligi Elektrik/Elektronik Bolum Kutuphanesi Tuzla - ISTANBUL / TURKEY	2
9.	LTJG. Sadi Karaman Runguc Pasa Mahallesi 121 Sokak No 8 Karacabey - BURSA / TURKEY	2
10.	Istanbul Teknik Universitesi Elektrik Fakultesi Dekanligi Istanbul / TURKEY	1

11. Orta Dogu Teknik Universitesi 1
    Elektrik Fakultesi Dekanligi
    Ankara / TURKEY

12. Bogazici Universitesi 1
    Elektrik Fakultesi Dekanligi
    Istanbul / TURKEY

13. Dokuz Eylul Universitesi 1
    Elektrik Fakultesi Dekanligi
    Izmir / TURKEY